Business-Policy driven Service Provisioning in HPC

Eugen Volk
High Performance Computing Center Stuttgart (HLRS)
Stuttgart, Germany
volk @ hlrs.de

Abstract—**Service provisioning in High Performance Computing (HPC) is typically defined in the way that implicitly corresponds to business policies of the HPC provider. Business policies, represented by business rules, objectives or directives, form means to guide and control the business of HPC service provisioning, affecting interdependently resource-management, SLA-management, contracting, security, accounting, and other domains. As business policies in HPC domain exist mostly implicitly, administrators configure resource management systems mostly intuitively and subjectively. This makes it hard for business people to assess whether business polices are consistent, and resource management behavior corresponds to business policies (and vice versa), as no linkage between business policies and scheduling policies is defined yet. In this paper we analyze relationships between business policies and resource management behavior, (1) presenting approach allowing to investigate how business policies and scheduling policies relate together, (2) identifying sources and key-factors influencing scheduling behavior, (3) describing relationships between those key-factors, and (4) using Semantics of Business Vocabulary and Business Rules (SBVR) for definition of business policies and transformation rules, capable to translate business policies into scheduling policies.**

Keywords - Business-Policy; Job-Scheduling; Policy-based Management; Policy-refinement; Business-Driven IT Management; SBVR.

I. INTRODUCTION

Service provisioning in High Performance Computing (HPC), reflected mostly by Job-Scheduling behavior, is typically defined in the way that implicitly corresponds to business policies of the HPC provider [1]. Business policies represented by business objectives, rules, or directives, form means to guide and control provisioning of HPC resources and services managed by a resource management system (RMS), which is responsible for resource management, job queuing, job scheduling and job execution. Business policies affect interdependently several domains involved in HPC service provisioning, such as SLA-management, contracting, resource-management, security, accounting, and others, and, might have direct or indirect influences on job-scheduling. For instance, a business policy, such as "all jobs of premium customers have to be started within 4 hours" has direct influence on scheduling, by determining the latest start of the job. In contrast, a business policy that demands "no violation of Quality of Services for platinum customers" may lead to higher priority of jobs of platinum customers, or to pre-allocation of resources dedicated to this customer's group. However, business-policies in HPC domain exist in most cases not explicitly, i.e., written by using domain specific language or natural language, but implicitly in the mind of the business people, whereas the configuration of resource-management-systems, in particular job-scheduler, is done by administrators.

The range of existing schedulers used for job scheduling in HPC varies from time based scheduler like Cron [4] to advanced policy-based schedulers like Moab [3] or its open-source variant Maui [2], which support large array of scheduling policies. Scheduling policies define thereby behavior of the scheduler by, i.e., assigning priority to a job depending on job-size (number of CPUs or cores required), estimated job-duration, user's priority and other factors. However, schedulers have a big amount of parameters and different scheduling policies which need to be selected and adjusted in order to meet business policies in different situations.

A problem occurs when administrators are configuring resource management systems, especially job-schedulers. The configuration of schedulers is done in most cases intuitively and subjectively, because of implicit business policies, system administrators unaware of them, or in general, because of missing link or mapping between business policies and selection and configuration of scheduling policies. This makes it hard for business people to assess whether current resource management behavior corresponds to business policies and vice versa, as no link between business policies and resource management is defined yet.

In our previous work [1] we presented approach allowing investigate how business policies influence scheduling policies. In this paper we apply this approach analyzing relationships between business policies and resource management system, (1) identifying sources and key-factors influencing scheduling behavior, (3) describing relationships between business policies, those key-factors and scheduling policies, and (4) using Semantics of Business Vocabulary and Business Rules (SBVR) for definition of business policies and transformation rules, capable to translate business policies into scheduling policies.

This paper is structured as follows. Section II presents related work in the area of job scheduling in HPC, SLA

based scheduling, policy-based management, business driven IT management, and semantics for description of business policies and business rules. Section III provides background information on job-scheduling in HPC. In Section IV we discuss the problem related to alignment of scheduling behavior with the business policies, showing the need for business-policy-based job-scheduling in HPC. Section V presents approach allowing investigating how business-policies relate to the job-scheduling in HPC and solve the problem described in previous section. Section VI analyzes influence of job-scheduling behavior on business metrics, identifying key-factors and relationships between scheduling policies, scheduling objectives, performance metrics and business metrics. In Section VII we analyze relationships between business metrics and business policies, identifying influences of business policies on job-scheduling, including relationships between different domains, such as SLA-management, Security, Accounting, etc. Finally, in Section VIII we present approach allowing expressing business policies in HPC using Semantics of Business Vocabulary and Business Rules (SBVR), providing examples for description of business policies. The last section summarizes this paper and outlines work in progress and future work.

II. STATE OF THE ART

In the target-area of "business-policy based resource-management/scheduling in HPC" currently no work is known to the author. However, there exists work in related areas, presented in the subsequent paragraphs.

Much research was done in job-scheduling in HPC domain, concerned on development and investigation of scheduling policies characterized by performance metrics, such as utilization, response-time, job-throughput, QoS violation, etc.. Feitelson et al. assessing several scheduling policies for parallel jobs, elaborating/identifying scheduling criteria and performance metrics [25][26]. Iqbal, Gupta and Fang [6] offer an overview about scheduling algorithms used for job-scheduling in HPC clusters. In [7], Casavant and Kuhl provide taxonomy of scheduling strategies in general-purpose distributed computing systems. In [8], Yeo and Buyya provide taxonomy of market-based resource management system, citing over 79 references. In [9], Abawajy describes recent advances in efficient adaptive scheduling policies. Achim Streit [27][28] investigated several job-scheduling policies for HPC dolmans, assessing their influence on utilization and response-time and developing adaptive-scheduling dynP algorithm which selects different scheduling policies, based on mean duration of all jobs in the job-queue. Proposed approach to "business policy based resource-management in HPC" uses scheduling criteria and performance metrics from job-scheduling in HPC area to elaborate linkage between scheduling policies, performance metrics and business level objectives.

In the area of SLA-based job-scheduling many papers have been published. SLA is part of a mostly short term service contract where the level of services or quality of services (QoS) is formally defined and agreed between service providers and customers. SLA contains usually rewards, for successful fulfillment of SLA, and penalties in case of SLA violations. SLAs are contracted in accordance with business-policies. Business-Policies prescribe kind of services and QoS which can be offered principally to the customer. Hence, SLAs can be considered as service level objectives contracted in accordance with the business-policies. On the other hand, business-policies are more prescriptive than SLAs, as SLA might be violated due to various reasons, but the behavior in a company must follow provider's business-policy. In [12][13][14][15], QoS and SLAs are used to find and allocate desired resources in quantity and quality, and determine priority and order of jobs for scheduling, among others, based on rewards and penalties declared in SLAs. In [13], authors describe how to derive IT management policies from SLAs, which in general follows autonomic computing approach (management by objectives).

Policy based management (PBM) aims at separation of rules and objectives governing the behavior of a system from its functionality [10]. Today, PBM is part of several management architectures and paradigms, including SLA-driven and business-driven management [10]. Many solutions in the area of policy-based management have been proposed since 1960 to present. In [10], Boutaba and Aib provide history of policy-based management, referencing over 118 papers. In IBM's autonomic computing reference architecture [11], the authors drafted the principle on how policies on high level might be used to express business needs/objectives that govern IT infrastructure operations. In IBM's Whitepaper [16] authors provide most recent definitions of policies and rules in business area, relating them to IT.

OMG's Semantics of Business Vocabulary and Business Rules (SBVR) [30] in its version 1.0, is recent (2008) standard intended to define "the vocabulary and rules for documenting the semantics of business vocabularies, business facts, and business rules" [30], used for the description of complex compliance rules. SBVR is interpretable in predicate logic with a small extension in modal logic, enabling consistency checking between rules. The proposed approach to "business policy based resource-management/job-scheduling in HPC" uses SBVR for the description of business vocabularies, facts, and policies.

Business Driven IT Management (BDIM) aims at a holistic management of enterprise IT infrastructure and services efficiently from business perspective [21], i.e., by aligning IT management decisions with business level objectives coming from the providers themselves, and their users [22]. Methods as used in BDIM are "based on mappings between IT technical performance metrics and business relevant metrics and exploit the linkage to provide decision support to IT management so as to maximize business value and IT-Business alignment" [21]. Existing work in this area [21][22][23][25] relates mostly to alignment of business requirements coming from business processes, with the management of IT-infrastructure of the enterprise systems, thus, addressing non HPC environment.

Moura et al. [21] provide a research agenda for BDIM, reviewing BDIM concepts and proposing a framework to assist in defining and describing BDIM usage domains. Sauvé et al. [25] present approach allowing to calculate business loss due to unavailability and high response time of web-services, aiming at minimizing costs and loss. In contrast to mentioned work in BDIM, the proposed approach to "business-policy based resource-management in HPC" is focusing on resource-management in HPC domain, in particular on job-scheduling. Whereas BDIM approach is mostly oriented on optimization of IT-Infrastructure to achieve business goals, proposed approach to "business policy based resource-management in HPC" is aiming at checking consistency between business policies themselves, and, between business policies and selection of job-scheduling policies in HPC domain.

III. BACKGROUND

Computing center infrastructure consists of several clusters used to provide computing resources to users. The cluster infrastructure of computing centers can be divided in two classes: high-throughput computing clusters and high performance computing clusters [6]. Nodes in high throughput computing clusters are usually connected by low-end interconnections. In contrast, more powerful nodes in high performance computing (HPC) cluster are interconnected by faster interconnection with higher bandwidth and lower latency. The application profile of high-throughput computing clusters includes loosely coupled parallel, distributed or embarrassingly parallel applications, requiring less communication and synchronization between nodes during the calculation. In contrast, the application profile of HPC clusters consists mainly of tightly coupled parallel applications, with high communication and synchronization requirements.

The computing nodes in cluster are managed by a resource management system (RMS), which is responsible for resource management, job queuing, job scheduling and job execution. Firstly, users who are willing to submit their applications or programs to resource management system need to express their applications as computational jobs, specifying requirements using, i.e., Job Submission Description Language (JSDL). Job specification contains usually number of nodes/CPUs/cores required, estimated maximum job-runtime, target architecture type (i.e., vector or scalar), specific I/O requirements (i.e., tools and files required for job execution) and other application- or platform specific parameters. After expressing application as a job, user submits the job in batch to queue of the resource management system, where it waits in the queue with the jobs of other users, until it is scheduled and executed. The wait time of the job depends on job-priority, system load, availability of requested resources and other factors [6]. Typically, a resource management system is comprised of a resource manager and a job scheduler [6]. Most resource managers have an internal, built-in job scheduler, which can be substantiated by external scheduler with enhanced capabilities [6], i.e., with support for various scheduling policies like Maui [2]. Resource manager provides scheduler with information about job-queues, loads on compute nodes, and resource availability. Based on that information, scheduler decides on how and when to allocate resources for job execution. The decision of the scheduler follows scheduling policy that determines the order in which the competing users' jobs are executed. The order of jobs typically depends on job-size (amount of resources, i.e., processors/cores required), estimated maximum job-runtime (indicated by user), resource access permission (established by administrator), resources available, and might depend additionally on QoS parameters (i.e., response time) expressed in contracts or SLAs.

The assessment of scheduling behavior is typically done according to various performance metrics [6][8][18][27][28], the most well known are:

- **Wait time**: the time a job has to wait before the execution of the job starts
- **Response time**: total time between when the job is submitted and when the job is completed. It includes wait time and execution time of the job.
- **Resource Utilization**: reflects the usage level of the cluster system - what percentage of available resources is used by jobs. Average resource utilization is calculated by total amount of resources/nodes used by all jobs within considered time period, divided by that time period.

A good scheduling policy is aiming at high resource utilization and short response times for the jobs, which are two conflicting goals. Typical performance criteria for users who expect minimal response time is the mean response time [6]. In contrast, administrators are typically trying to achieve maximum overall resource utilization, as that maximizes profit. Improving overall resource utilization and at the same time decreasing mean response time are two conflicting goals, as short waiting times are achievable only with low utilization [26]. Typically, scheduling policies that optimize resource utilization prefer those jobs which need many resources (large jobs) over long time period (long jobs) [26]. However, this causes that short jobs requesting few resources need to wait longer until long and large jobs are finished. Contrary, scheduling policies preferring short and small jobs would reduce the average response time [26]. Because job-size and job-length are varying from job to job, as well as the job-submission rate, gaps in schedule occurs, which are reflected by utilization drop [30]. The challenge in design of scheduling policies is to find tradeoff between optimizing these two (and other) mostly contradicting performance metrics [26]. This tradeoff should be derived from the business demands or business level objectives, expressed in business policies. Hence, there is a need for a preference specification, making tradeoff between contradicting performance metrics, derived from business objectives or demands.

IV. NEED FOR BUSINESS-POLICY-BASED JOB-SCHEDULING

Business policies are control statements that guide behavior in a company and control the business. Business policies are defined usually at an overall strategic level influencing and controlling various areas participating in business provisioning by setting business level objectives, rules and other constraints. In HPC domain, these areas are: security, contracts and SLAs, resource management, accounting, and others. Business policies, which relate to security, contain statements governing the access to HPC resources, i.e. prescribing the process of obtaining permission to HPC resources, granting, restricting or refusing the access, taking external regulation into account. Contract and SLA business policies describe the spectrum of HPC services offered principally, including Quality of Services (QoS) and capacity capabilities. Resource management business policies contain statements influencing resource allocation and scheduling behavior on a high level, by, i.e., prescribing the preferences between users-groups, tradeoffs between different performance metrics, scheduling optimization criteria, resource allocation strategies, and others.

As already mentioned, scheduling behavior is typically defined in the way that it implicitly adheres to business policies of the HPC providers, while taking users' job requirements, available resources, existing SLAs, long term contracts and other factors into account [1]. Advanced policy-based schedulers like Maui [2] have a big amount of parameters and different scheduling policies which need to be selected and adjusted in order to meet all business policies in different situations. As business policies exist mostly implicitly in the mind of people, or, because administrators are not really aware of all of them and their interrelationship with site-effects, administrators configure schedulers mostly intuitively and possibly subjectively. This makes it hard for business people to assess whether the actual scheduling behavior is correct and corresponds to current business policies, as there is no linkage between business policies and scheduling policies defined. Additionally, there might be new business policies, or a fast switch between different business policies required, affecting job-scheduling. Hence, for right configuration of job-scheduling behavior it is essential to understand:

- What are the business requirements that affect job-scheduling?
- Where are these requirements coming from?
- How are they influencing job-scheduling?
- Are there any conflicting requirements?
- What is/should be the tradeoff between conflicting requirements?
- On what is this tradeoff dependent?

For instance, in profit oriented organizations, managers try to achieve maximum profit, which often means that they deliver various quality of services (with specified expected response-time) to various users and groups [2], aiming at increasing system utilization at specified expected response time level. In contrast, nonprofit organizations, like computing centers at universities are delivering HPC resources to various users and groups on best effort basis, neglecting response time, thus, focusing only on the overall resource utilization to increase amount of jobs completed. Some of the national computing centers affiliated to universities have joint collaboration with scientific and industrial partners through common joint cooperation company, offering HPC services with certain QoS level. That means the scheduling behavior in clusters of such computing centers needs to be adapted to various business needs, even at the same time, leading to differentiation between at least two different QoS service classes, e.g., silver class with specified expected response time, and, bronze class with best effort. Such requirement could lead to a higher prioritization of jobs of industrial users, comparing to jobs of students or scientific users. In addition to silver, and bronze, there might be gold service class offered for urgent computing, whose jobs are scheduled immediately preempting other jobs.

Furthermore, there are cases where the usual job-scheduling behavior must be adapted to changing situations and require evaluation of several business polices. Assuming there are at least two separated clusters – one for industrial users and, one for research users and students. In case of fall-out of the cluster on which jobs of industrial users are executed, these could be shifted to another cluster, if allowed. The answer on the question whether the jobs of industrial users might be shifted, e.g., to research cluster, on which jobs of students or researchers are executed, depends thereby on evaluation of several business policies and constraints. Research and educational clusters are typically financed by federal authority, whereas clusters used for industrial calculations are financed through common joint cooperation company. In case of the business policies, which prescribe that (1) industrial partners have higher importance than students or researchers, (2) only the owner (who has financed it) of the cluster decides on permissions, and (3) current federal land policy that impose to use research clusters only by researchers or students, then the shifting of industrial jobs to a research cluster is not allowed. Alternatively, if a business policy prescribes that (1) industrial partners have higher importance than students or researchers, (2) only the service provider decides on permissions, then shifting of industrial jobs to research cluster is allowed, it is even an obligation.

As stated, there are many different business policies from different areas, which need to be considered when configuring schedulers. Furthermore, there might be a fast switch between different business policies required, and a fast adaptation of the scheduling behavior dependent on evaluation of several business policies from different domains. Because of implicit existence of business policies and missing link between business policies and scheduling policies, there is a risk of resulting incorrect scheduling behavior.

V. APPROACH

An approach to handle problems described in previous section, induced by changing business objectives or altering situations, might follow IBM's autonomic computing

reference architecture [11]. Autonomic computing is thereby defined "as a computing environment with the ability to manage itself and dynamically adapt to changes in accordance with business policies and objectives" [11]. Following this approach, there must be (1) business policies defined, capable to express business requirements influencing scheduling behavior on high level. Once, there are business policies defined, the next step (2) consist then of transforming these business policies with other sources (as SLA, Contracts, Accounting, etc.) influencing scheduling behavior into scheduling policies to configure advanced policy-based schedulers like Maui or Moab. In order to define business-policies explicitly, there must be HPC business policy specification language elaborated, capable to express business needs for various situations. In order to address this problem, we will follow a bottom-up process:

The first step (1) consists of the analysis of existing scheduling policies in HPC in order to identify: scheduling criteria used by scheduling policies, scheduling objective function which is approximated by scheduling policies, and performance metrics/indicators characterizing costs of scheduling. The first step includes also identification of relationships between the elements of scheduling. The results of the first step are described in Section VI.A. The next step (2) involves the analysis of performance metrics and their influence on business, metered by business metrics. Section VI.B presents results of the second step, describing business metrics and relationship to scheduling performance metrics. The outcomes of the first and second step are summarized in third step (3) as a model, presented in Section VI.C, identifying key-factors and their relationships influencing scheduling behavior, taking business metrics, scheduling performance metrics, scheduling policies, user-requirements, SLAs/contracts and resource-capacities into account. In the next step (4) we identify influence of business policies on business metrics and scheduling behavior, considering various sources of influence. The outcomes of the fourth step are summarized in Sections VII.A and VII.B, presenting relationships between business metrics and business policies, identifying sources of influence on job-scheduling behavior coming from various domains, including SLA-Management, Resource-Management, Security/License Management, etc. Especially the relationship between existing business policies, business metrics and scheduling policies will provide an overview on how policy refinement process of transforming business policies to scheduling policies might principally looks like. In the fifth step (5) we propose usage of Semantics of Business Vocabulary and Business Rules (SBVR) for description of business policies and transformational rules, capable to provide semantic framework for definition of vocabulary and business policies/rules (including business policy schema) used to describe business policies, consistence checking rules and transformational rules. Results of the fifth step are described in Section VIII, presenting examples describing business policy schema and transformation of business policies into scheduling policies using SBVR. Thereby we present examples for business policy schema, enabling description of business policies.

Finally, in order to evaluate results achieved in previously steps, the last step consists of the reference implementation, enabling mapping of reference business policies together with other key factors to scheduling policy configuration for advanced schedulers such as Moab [3] or Maui [2].

VI. From Job-Scheduling-Policies to Business-Metrics, and Back

In this section we analyze the influence of job-scheduling behavior on business. Firstly, we analyze job-scheduling policies to identify relationship between scheduling criteria, such as job-size, job-length, etc., and scheduling performance metrics, such as utilization, response-time, and others. In the second section, we analyze influence of scheduling performance metrics on business, identifying business metrics and their relationship to performance metrics.

A. Analysis on Job-scheduling in HPC

Job-Scheduling algorithms can be divided in two classes: time-sharing and space-sharing [6]. Time sharing algorithms divide time on a processor into several slots, each time-slot is assigned to unique job then. In contrast, space-sharing algorithms assign requested resources exclusively to unique job, until job is completed. In all HPC clusters is space-sharing approach used, as time-sharing approach increases synchronization overhead between nodes of the same job.

According to Streit [28], Resource Management Systems can be divided into queuing and planning systems, depending on their planned time frame. Queuing systems try to utilize currently free resources; in contrast planning systems are not restricted to present time, but take also future into account assigning resource-reservation to future jobs. According to Feitelson and Rudolph [26], queuing systems can be classified into on-line vs. offline, with online subdivided into closed and open models. Off-line model assumes that all jobs are available from the closed set of jobs, with no later arrivals. In contrast, on-line model assumes that jobs arrive over period of time. A closed on-line model expects fixed set of jobs to be handled; in contrast, open on-line model is characterized by endless stream of jobs [26]. In HPC centers, open online model is most commonly used, as it reflects real user-behavior – continuous submission of jobs as a stream. Offline models are used as well, for the processing of batch jobs during the night or weekends.

Massive parallel systems as used in HPC are typically operated in following way [25][28]: The system is divided in partitions on which parallel jobs are executed. Thereby, a partition consists of several nodes assigned to one or several job-queues. The partitioning can be done according to job-characteristics (i.e., job-size, job-length) and priorities (i.e., high-priority, best effort) [28]. Within a job-queue several

scheduling policies can be applied, FCFS is the most commonly used scheduling-policy. Nodes of a partition are assigned to different possible prioritized queues. Jobs in prioritized queue are taken first to be executed on free resources. Once a job is started with the requested number of nodes, it is running until the job is completed [28].

According to Feitelson et al. [25] in [28], jobs can be classified into rigid, moldable, evolving and malleable jobs. Rigid and moldable jobs are depending on whether the number of assigned resources to a job is decided by the scheduler (moldable) or is fixed by the user at start-time (rigid). Evolving jobs arise when applications go through distinct execution-phases with changing amount of resources, resulting in allocation of required resources in each execution step [25]. Malleable jobs are those which are capable to deal with changing system capacities, resulting in an increase of decrease of resources to be used by a job. The rigid jobs with fixed amount of required resources are the most difficult for scheduling. In HPC, rigid jobs are mostly used. As mentioned before, jobs have various requirements, differing on quality (Memory, CPU-speed, IO-bandwidth/latency), quantity (amount of resources – typically number of cores/nodes) of resources and expected run-time. However, the job-runtime is only estimated, as the actual job-run-time depends on application-characteristics such as number of nodes-used, speedup-factor, whether the job is memory, CPU or IO bounded, and machine characteristic on which jobs are executed.

According to Krallmann, Schwiegelshohn, and Yahyapour [32] in [28], scheduler can be divided in three parts: a **scheduling policy** determining allocation of resources to jobs; **objective function** describing the cost of the complete schedule, such as response-time, utilization, job-throughput, etc., **scheduling algorithm** generating valid schedule according to objective function.

As optimal scheduling is NP hard problem, it requires high computational effort to calculate perfect schedule. Approximation to optimal scheduling can be found in polynomial time, i.e., using specific heuristics as outlined below.

Job scheduling policies have two important phases [28]:

1. Putting jobs in the queue at submit time
2. Taking jobs out of the queue at start time

Putting jobs in the queue can be done by sorting jobs according to:

- Arrival time (FCFS) – the most known scheduling policy with fairness. The jobs that arrive later start later.
- Increasing estimated job-runtime (SJF). This strategy is aiming at minimizing mean response time. However, SJF is not fair strategy as longer job may be starved - failed to be scheduled.
- Decreasing estimated job-runtime (LJF). This strategy is aiming at resource utilization, as it acquires resources for longest possible time period hence for longest job. LJF is not fair, as shorter jobs may be starved.
- Increasing or decreasing number of requested resources. Decreasing strategy is aiming at maximizing resource utilization, as it acquires as much resources as possible, as required by the largest job. Increasing strategy is aiming at minimizing
- Increasing or decreasing used area of the job (estimated runtime * requested resources). Decreasing strategy is aiming at maximizing utilization, as it tries to allocate as much cpu-time for job as possible. Increasing strategy is aiming at minimizing mean response time while maximizing utilization.
- Increasing time to deadlines (EJF). This strategy is aiming at satisfying deadlines, to prevent any violation of contracts or SLAs.
- Given Job-weights: the higher the weight the higher the priority of the job is. Higher prioritized jobs are executed before the lower jobs. The job-weight can be based on customer importance, granting important customers' shorter response-time.
- By the Smith ratio, that is defined as a ratio between job-weight and its area (run-time * required resources).
- And many other scheduling policies based on other criteria, and their mathematical or logical combination.

Taking jobs out of the queue can follow different strategies – here some examples [28]:

- "Always start the head of the queue." Lack of resources to serve head job leads to delay of the other jobs, even if there are enough resources available. This approach is called front.
- "Search the queue from the beginning and take the first job that can be started immediately fitting given constraints" [28] - that fits into current schedule. This strategy is called FF – fist fit. It tries to optimize resource utilization, but has a drawback that in worst case a job can wait forever.
- Search the queue from the beginning and take the job that can be started immediately and leaves latest resources free. This strategy is called BF - best fit.
- Combination of several of these and other strategies is used to optimize and improve scheduling.

During the schedule there might be gaps between jobs– reflecting idleness of resources and indicating drop in resource utilization. In order to fill out these gaps and optimize utilization, two backfilling strategies exist [28]:

- Conservative backfilling selects only those jobs to fill out the gap, that are not delaying other waiting jobs in the queue. This is a predictive strategy that ensures fairness and increases utilization.

- EASY backfilling is more aggressive and selects those jobs to fill-out the gap that are not delaying the waiting-queue head. This is less predictive strategy, since only the scheduling of the head jobs is assured. In worst case, jobs may be starved.

Hence, simple scheduling algorithms might be enhanced by combining them with the use of advanced reservation and backfill techniques. Advanced reservation algorithms, as used in planning system, use estimated job-runtime to make reservation on resources for particular jobs and create time-schedule for certain time period. The problem thereby is that schedule is based on estimated job-runtime, which is in most cases much longer than the real one. That means the schedule needs to be adapted as soon as jobs are completed earlier than expected. The backfilling strategy improves basic strategies by combining them with additional iteration to fill out the gaps, as outlined previously. Given schedule on high priority jobs i.e., by applying LJF strategy, the scheduler use in second iteration lower priority jobs to fill out the gaps (free time slots on unused resources) between higher priority jobs.

As mentioned, the assessment of scheduling behavior is done according to various performance metrics. However the selection of right performance metrics depends on system type: open online/offline and closed. In addition to metrics presented in Section III, such as wait-time, response-time and resource-utilization, there exists various other metrics, depending on type of queuing system [26]:

- **Makespan:** total time for the completion of all jobs. It is a metric for offline queuing systems, since the number of jobs in an online open model is assumed as infinitely.
- **Throughput**: number of jobs completed in a period of time. It is a good metric for closed systems, with fixed number of jobs.
- **Average Response Time:** is ratio between the sum of all jobs' response-time and the number of jobs (or total time for waiting and execution of all jobs divided by the number of jobs). It is widely used for open online systems. However, this metrics seems to make emphasis on long jobs, as opposed to shorter jobs which are most common [26]. A possible solution is normalization of the response time by slowdown.
- **Slowdown:** ration between response time (wait-time + running time) and running time. Hence, the slowdown is the response time normalized by the running time. The problem with slowdown is that extremely short jobs with even acceptable wait-time lead to high slowdown. Hence there is a need for boundaries.
- **Bounded Slowdown:** is slowdown by applying lower bound to job-runtime.
- **Loss of capacity:** as opposite to utilization, loss of capacity determines how much percent of all resources were idle despite of jobs waiting in the queue.

Additional metric, which play essential role on rewards or penalties is the **missed deadline** metric for each job. As in case of SLAs, not meeting deadline for a job may result in penalties, thus influencing the profit function of the provider directly.

As mentioned previously, performance metrics can be divided into user centric metrics, and provider centric metrics. Provider centric metrics are focusing on resource utilization, throughput, makespan etc. In contract, user-centric metrics refer to actual job-performance, relating mostly to wait-time, response-time, average-response-time, slow-down, etc.. However, providers are interested in user-specific metrics as well, to ensure that the level of quality of services as requested by users is achieved, to satisfy users. At the same time, users are interested in utilization metrics as well, as they know that underutilized system has typically short response time, as presented in Figure 1.

Hence, performance-metrics are trying to formalize scheduling goals [25]:

1. Satisfy the user
2. Maximize profit

User satisfaction can be achieved by reducing the response time [25], however at the price of reduced load, as shown in Figure 1. Using open online queuing model implies that the scheduler has to deal in worst case with extreme situation [26]. The analysis on scheduling policies metered by, i.e., response-time and utilization, tries to find out when the utilization breaks down, because of high system load [26], as shown in Figure 1.

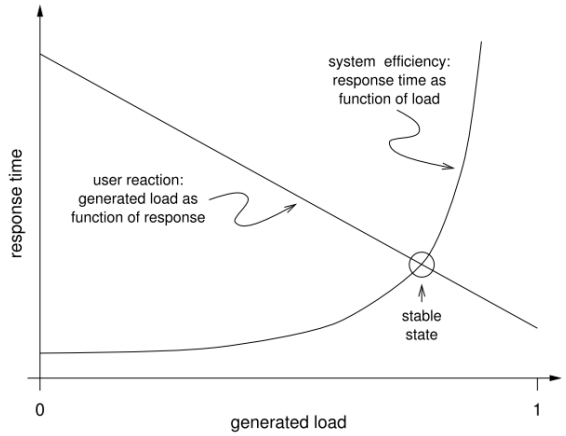

Figure 1. Response time vs. load [26].

In order to achieve certain QoS level, advanced reservation protocol may be used, that reserves and allocates required amount of resources for certain time period, ensuring meeting latest deadline of the job-execution [27].

The scheduling of other jobs arriving at allocated time-period has to deal with the reminder of available resources.

In addition to scheduling policies, that determine the priority of jobs based on objective, there are fairness policies that are managing usage on resources between different user-groups and jobs. Fairness implies giving all users equal access to resources [2]. However, different concepts incorporating historical resource usage, political issues, and job value are equally valid, depending on the preferences of the provider; as example, here a list of fairness policies supported by Maui [2]:

- **Throttling polices** specify limits on resource usage for a jobs, a user-group or a project.
- **Job-Prioritization policies** allow to balance between different performance metrics such as response-time, utilization and other, by assigning weighting factors or base priority to different service classes.
- **Fairshare policies** are used for job feasibility and priority decisions, by limiting resource access or adjust priority based on historical resource usage by users/groups/QoS-classes/queues [2].
- **Allocation policies** specify long term, credential-based resource usage limits. Resource allocation policies grant a job a right to use a particular amount of resources at particular time-period. Limits might be applied to particular machines, or globally usage, containing activation and expiration date, as well the amount of granted resources [2].

To summarize, scheduling policies determine priority of jobs based on various criteria, such as arrival-time, job-runtime, job-size, selected level of quality of services (QoS), taking limitations such as fairness, fairshare and allocation policies into account. The assessment of schedulers is done according to performance indicators. The performance of scheduling is reflected by several performance indicators, such as response-time, system-utilization, missing deadlines, etc. The selection of right performance indicators depends on the queuing model, and, in particular on objectives given by the provider derived from its business demand. Providers have usually multiple goals which may include maximizing resource utilization, ensuring certain level of QoS reflected by response-time, giving preference to certain customer/user-groups/project, etc.. However these goals are mostly conflicting and require tradeoffs, claiming their relative importance to each other, while taking certain constraints into account. In next section we describe relationship between scheduling and business.

B. Scheduling and its Influence on Business

As mentioned in the previous section, scheduling performance-metrics are trying to formalize business goals, which might be, i.e., maximize profit, satisfy the user, increase own reputation, etc. In this section we analyze influence of scheduling performance metrics/indicators on business goals such as profit and user-satisfaction, identifying relationship between business-goals, business metrics and performance metrics. Following subsections provide mathematical definition of business metrics.

Profit is defined as a difference between total revenue and total costs:

$$Prof_{total} = Rev_{total} - C_{total} \quad \text{(F1)}$$

A Revenue Rev_{ijd} for executing a job *j* on computing resources of type *i* (homogenous cluster with machines of type *i*) with required number of nodes n_j, execution time e_{ij} and deadline *d* can be defined as follow (based on notation as used in [33]):

$$Rev_{ijd} = e_{ij} * n_j * p_{id} \quad \text{(F2)}$$

with

e_{ij} - execution time (hours) of job *j* on resource of type *i* reflects different execution time of the same job on different machines, dependent on capabilities provided by resources (CPU-speed, memory, IO-interconnect)

n_j – number of CPUs required for job *j* reflects requirements of the rigid (fixed amount of CPUs) job

d_j – QoS class of job *j* with deadline *d,* reflecting expected response time

p_{id} – price per CPU-hour of machine-type *i* for QoS class with deadline *d,* reflecting different prices for different machine-types and different QoS classes (expected response-time). In reality, even the same QoS class on the same machine can have different prices for different customer groups. For example, some national supercomputing centers have two different prices – one for researchers and one for industrial users, due to government grant aiming at supporting researchers.

Considering the revenue formula, it must be noted that the product of $e_{ij} * n_j$ indicates resource-usage of job *j* on cluster *i*, while meeting QoS requirements *d*. Hence, contribution of job j to utilization on cluster i is:
utilization = $e_{ij} * n_j$ divided by total number of resources in cluster *i*.

Total revenue Rev_{total} can be defined for: various QoS classes *d* (1…D), jobs *j* (1,..,J), and clusters *i* (1,..,N) as follow:

$$Rev_{total} = \sum_{d}^{D} \sum_{j}^{J} \sum_{i}^{N} Rev_{ijd}\, x_{ijd} \quad \text{(F3)}$$

with

$$x_{ijd} = \begin{cases} 1, & \text{if job } j \text{ of QoS class } d \text{ was executed on machine type } i \\ 0, & \text{otherwise} \end{cases}$$

Costs for executing a job j on a resource-type *i* can be obtained as follow:

$$C_{ij} = e_{ij} n_j c_i \quad \text{(F4)}$$

with:

c_j – costs (€ per hour) for executing a job on machine-type i . This function contains electricity costs per hour, as well as machine specific hardware and software costs averaged by usage period (which is in HPC domain usually 3 – 5 years). Thereby, costs for the job-complexity, as a result of job behavior reflecting CPU, IO and memory usage are implicitly covered as average values.

n_j – number of CPUs required for job *j*

e_{ij} - execution time (hours) of job *j* on resource-type *i*

Total costs can be defined as a combination of variable costs C_{ij} and fixed costs C_{fix}, containing maintenance, software, facility, and other fixed costs.

$$C_{total} = \sum_{j}^{J} \sum_{i}^{N} C_{ij} x_{ij} + C_{fix} \quad \text{(F5)}$$

The profit model presented reflects mainly long term contracts, where rewards for meeting QoS requirements are defined as Revenues. Penalties in long term contracts are usually not reflected monetary, but may lead to customers leaving providers, resulting in lower load and system utilization. In contrast, short term contracts expressed as SLA use monetary rewards and penalties associated with fulfillment and violation of SLAs demanding certain level of QoS (response-time or deadline).

Rewards may be defined in a similar way as Revenues for executing job *j* on machine *i* while meeting deadline. Although, other price models exists, where fixed rewards are paid, independent on used cpu-time, or, are proportional to difference between actual response-time and contracted response-time, as presented by Abraho et al. [34].

Penalties in SLAs are reflected monetary, expressing the price as a fixed value (per cpu-time) or as a function of violation degree. The higher the excess between contracted and actual QoS level (response-time or deadline) is, the higher the penalty (per CPU-hour or fixed) is paid, as presented by Abraho et al. [34] in Internet data center service domain, where requests for same application class of QoS have same capacity demands, but variable arrival rate.

The influence of Rewards (Rew) Penalties (Pen) on Profit can be expressed simplified as follow:

$$Prof_{total} = Rev_{total} - C_{total} + Rew_{total} - Pen_{total} \quad \text{(F6)}$$

with:

$$Rew_{total} = \sum_{d}^{D} \sum_{j}^{J} \sum_{i}^{I} Rew_{ijd}\, x_{ijd} \quad \text{(F7)}$$

$$x_{ijd} = \begin{cases} 1, & \text{if job } j \text{ of QoS class } d \text{ was executed} \\ & \text{on machine type } i \text{ meeting deadline } d \\ 0, & \text{otherwise} \end{cases}$$

Rew_{ijd} as a Reward function expressing monetary value for meeting contracted QoS, as declared in SLAs, see [34] for details.

$$Pen_{total} = \sum_{d}^{D} \sum_{j}^{J} \sum_{i}^{I} Pen_{ijd}\,(1 - x_{ijd}) \quad \text{(F8)}$$

Pen_{ijd} as a penalty function expressing monetary value of violating contracted QoS, as declared in SLAs, see [34] for details.

In order to minimize violation on SLAs, it is necessary to determine available capacities for each level of QoS offered. Existing work on capacity planning in Grid, as presented by M Siddiqui, A. Vallization, T. Fahringer in [35] introduced new mechanism, based on advanced co-reservation, that optimizes resource utilization and QoS constraints among grid resources. In order to achieve certain QoS level, advanced reservation approach reserves and allocates required amount of resources for certain time period, ensuring latest deadline and implicitly start of the job-execution.

The presented profit function provided simplified linkage between utilization and profit function, reflecting response time as rewards and penalties, depending on violation of contracted QoS levels. In addition to profit business metric, other finance metrics such **Return on Investment (ROI)** can be used to measure value of the HPC system:

$$ROI = \frac{Rev_{total}}{C_{total}} \quad \text{(F9)}$$

Traditionally, HPC systems have been valued according their utilization; but this lead to equal treating of problems, jobs of different complexity and purpose, independent of their business value for the organization, and possibly not optimizing users' needs [36]. Without considering these issues, the investments on hardware, software, and other upgrades, i.e., aiming at energy-efficiency, appear to be blindly, not aiming at optimizing users' need and productivity of the system [36]. To overcome these issues, ROI, expressed by **Benefit-Cost Ratio (BCR)** calculation, can be used to value system according their benefit. BCR is defined as "profit or cost savings divided by the sum of the investment over a given time period" [36]. Thereby, the time period for renewing of hardware/software etc. in HPC domain is usually 3-5 years.

$$BCR = \frac{benefit}{cost} \quad \text{[36] (F10)}$$

BCR is similar to classical definition of ***productivity***, as ratio between utility and costs [36]:

$$productivity = \frac{utility}{cost} \quad \text{(F11)}$$

However, the definition of benefits/utility and costs depends on organization type, that uses HPC. For example for a research-oriented institution like a university or national laboratory, HPCS productivity model [36] defines utility/benefit as a function on "time saved by engineers or researchers in solving advanced problems", taking into account not only the system costs, but also time on parallelization, training, launching and administration [36]:

$$\underset{(BCR)}{Productivity} = \frac{\sum (\text{time saved by users on system})}{\left(\begin{matrix}\text{time to}\\ \text{parallelize}\end{matrix}\right) + \left(\begin{matrix}\text{time to}\\ \text{training}\end{matrix}\right) + \left(\begin{matrix}\text{time to}\\ \text{launch}\end{matrix}\right) + \left(\begin{matrix}\text{time to}\\ \text{administrate}\end{matrix}\right)} \quad \text{(F12)}$$

For industrial organization, where HPC systems are used mostly for solving product design and development challenges, industrial users are concerned mostly on value of the product, its market-share, resulting profits generated, etc., leading to assessment of importance of jobs or projects associated that value. Hence, the BCR can be defined as follow [36]:

$$\underset{(BCR)}{Productivity} = \frac{\sum (\text{profit gained or maintained by project})}{\left(\begin{matrix}\text{cost of}\\ \text{software}\end{matrix}\right) + \left(\begin{matrix}\text{training}\\ \text{cost}\end{matrix}\right) + \left(\begin{matrix}\text{admin}\\ \text{cost}\end{matrix}\right) + \left(\begin{matrix}\text{system}\\ \text{cost}\end{matrix}\right)} \quad \text{(F13)}$$

This definition is valid as well for HPC provisioning, where importance of different projects is equivalent to importance of different customers (users-groups), or in general to importance of jobs in scope of job-streams (with varying job-size, job-length and job-complexity) of different QoS classes executed on different machines.

As costs and speed varies from machine to machine, how does a faster machine influence on user-behavior, profit and price?

The answer on this question can be explained by Amdahl's Law:

$$T(N,p) = \left[\left(\frac{1-q}{p}\right) + q\right] * T(N,1) \quad \text{(F14)}$$

With:

q Sequential fraction of the program

(1-q) parallel fraction of the program

p number on processor-cores

T(N,1) – Time for sequential execution for a problem size N using best sequential algorithm

T(N,p) – Time it takes to solve a problem of size N on p processors using best parallel algorithm

This is equivalent to

$$T(N,p) = \frac{T(N,1)}{S_p(N)}$$

with Speedup expressed as:

$$S_p(N) = \frac{1}{\left(\frac{1-q}{p}\right) + q} \quad \text{(F15)}$$

This leads to following conclusion:

1. The faster a machine is, the faster a job can be executed on it. Doubling the computing-speed on each core (on CPU), leads to halving the computing time (assuming the same Speedup). At the same time this leads to halving the response-time, as more jobs can be processed.
2. The longer a job is, the greater is its time-saving potential on faster machine.
3. The larger a job is, the shorter is its response-time (wait-time and execution-time) on faster machine, as the capacity of the machine increases with its speed.

Thus, a user would rather prefer a faster machine for long and large jobs, even if the prices are higher.

The next question arises on **prices** (per core/CPU hour) between two machines-types *A* and *B*. To calculate possible price-range on different machines, we need to calculate:

a) the lowest price per CPU-core-hour, the provider can offer to cover the exploitation-period of the machines, with particular assumption on average utilization for planned time-period

b) the highest price per CPU-hour, reflecting the value for the user.

The lowest price per CPU-core-hour can be calculated by dividing TCO (Total cost of ownership), including costs for software, maintenance, administration, and systems, by exploitation-period, number of CPU-cores and expected utilization:

$$price_{low} = \frac{\left(\begin{matrix}\text{cost of}\\ \text{software}\end{matrix}\right) + \left(\begin{matrix}\text{training}\\ \text{cost}\end{matrix}\right) + \left(\begin{matrix}\text{admin}\\ \text{cost}\end{matrix}\right) + \left(\begin{matrix}\text{system}\\ \text{cost}\end{matrix}\right)}{\left(\begin{matrix}\text{exploitation}\\ \text{period in hours}\end{matrix}\right) * \left(\begin{matrix}\text{number}\\ \text{of CPU\_cores}\end{matrix}\right) * \left(\begin{matrix}\text{expected}\\ \text{utilization}\end{matrix}\right)} \quad \text{(F16)}$$

The highest price per CPU-hour depends on its value for the user. We calculate the price relative to the slower machine. We define initially, the value of the job *j* as the Revenue obtained by executing job *j* on *p* cores of machine-type *i*, with job duration $T_{ij}(N,p)$ and price p_i:

$$Value(j_{pi}) = Rev_{ij} = T_{ij}(N,p) * p * p_i \quad \text{(F17)}$$

For simplification, we demand:

$$Value(j_{pA}) = Value(j_{pB}) \quad \text{(F18)}$$

Thereby, the value of the time-savings from the user-perspective is not taken into account.

$$T_{Aj}(N,p) * p * p_A = T_{Bj}(N,p) * p * p_B \quad \text{(F19)}$$

$$\frac{p_A}{p_B} = \frac{T_{Bj}(N,p)}{T_{Aj}(N,p)} = \frac{T_{Bj}(N,1) * S_p(N)}{T_{Aj}(N,1) * S_p(N)} = \frac{T_{Bj}(N,1)}{T_{Aj}(N,1)} \quad \text{(F20)}$$

$$p_A = p_B * \frac{T_{Bj}(N,1)}{T_{Aj}(N,1)} \quad \text{(F21)}$$

Hence, the price per core-hour on the machine-type A is in ideal case (neglecting the effect of time-saving) proportional to the speed factor on the machine-type B. Taking the time saving on job-execution into account, the value on time saving can be adapted as follow:

$$Value\left(T_{Aj}(N,p)\right) = Value\left(T_{Bj}(N,p)\right) + Value\left(T_{Bj}(N,p) - T_{Aj}(N,p)\right) \quad \text{(F22)}$$

However, value of time-saving is rather subjective and hard to reflect monetary; it depends on the purpose of the job as mentioned previously.

To summarize, increasing the speed on a CPU-core, leads to higher preference of the cluster for users with large and long jobs, despite to higher CPU-core-prices. This leads to better utilization-potential of the cluster. At the same time, increasing CPU-core speed, leads to higher capacity of the cluster, assuming the number of cores is at least the same. However, current trend on CPU design is focusing increasingly on energy efficiency, leading to increasing number of cores per CPU, in contrast to increasing CPU-core-speed consuming more energy.

In conclusion, in this section we defined business metrics and described relationship between performance metrics, such as utilization, response-time, and business metrics, including productivity, profit, revenue, costs, rewards, penalties and prices, taking CPU-core-speed and user behavior into account.

C. *Identifying Key-Factors and Relationships*

Analyzing the scheduling algorithms and policies leads to identification of key-factors, characterizing (and determining) scheduling behavior and influencing business. In this section we identify Key-Factors and summarize their relationships, as shown in Figure 2, according to explanation provided in previous sections.

Scheduling is a function which is aiming at optimizing resource-allocation (assigning available resources to jobs), while ensuring that requirements of users/customers are satisfied according to objectives of the provider.

Customer requirements are expressed as capacity requirements, jobs specifying job-size (amount of resources) and job-length (runtime of the job), and as QoS requirements, called also as Service Level Objectives (SLO), metered by customer specific metrics such as response-time, deadline-missed, etc. However, capacity requirements as contracted in contracts or SLAs between the provider and a customer specify mostly the estimated total capacity (CPU or core hours) to be used within a period of time, thus hiding the nature of jobs, in particular job-size, job-length, job-submission-time, job-submission-rate, making it impossible to create optimal scheduling-plan for resource-allocation before the arrival of jobs. This makes it hard for providers meeting required QoS level, in case there are not enough resources available to satisfy demand of all jobs waiting in the job-queue.

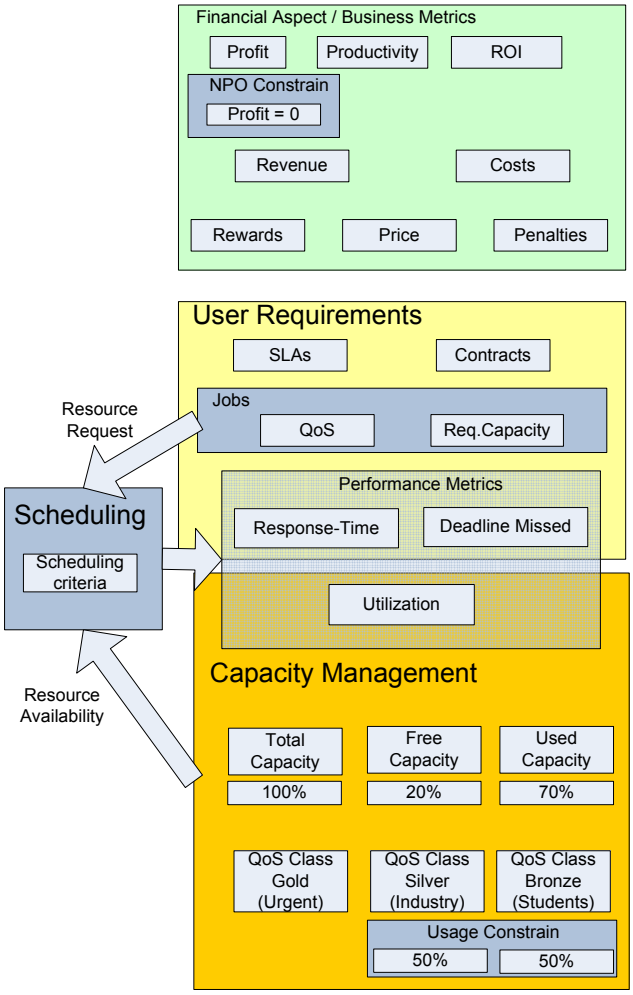

Figure 2. Scheduling and its influence on HPC service provisioning.

The **objectives of the provider** (Business Level Objectives) are depending on the purpose of the organization (his/her mission), his/her preference on profit-orientation or user-satisfaction, external influence of regulations and policies, and other factors. The profit-orientation of the provider and customer-satisfaction are metered by **business metrics** presented in previous section, using profit, ROI, productivity, revenue, costs, rewards, penalties.

The relationship between business metrics and scheduling performance metrics was explained in previous section. As pointed out, the profit is dependent on revenue,

costs, rewards and penalties. Revenue is dependent on system utilization (the higher usage of resource, the higher is the revenue) and price (per cpu-hour), which is dependent at least on system capacity and capability. Penalties are dependent on fulfillment of contracts and SLAs, as metered by performance metrics – exceeded response-time and missed deadlines. Penalties may result not only in monetary payment, but also in loss of customers/users, leading to decrease on utilization. The loss of customers might lead to loss of company-image, resulting in further decrease on utilization for a long time-period. In order to minimize the risk on violating contracts or SLAs, the provider needs to make planning on capacities.

In order to realize **capacity management** which purpose is to plan allocation of capacities to different QoS levels, a provider has to know the saturation point of the system. This saturation point is determined by system-load (system-utilization), scheduling policies, and accepted response-time, as pointed out by Feitelson et al. [26], and Streit [28], shown in Figure 1. However, the system load is hard to predict, as job-stream is varying in job-size, job-length, job-submission-time, job-arrival-rate, and is based on user-behavior. To meet QoS requirements, it's necessary: to make prioritization between jobs of different QoS levels, to use fair-share policies, regulating the usage of capacity between several customers and QoS levels, by adjusting priority according to the usage-history, or to partition the system according to capacity allocated for each QoS level. However, there is still a danger of not meeting QoS, resulting in penalties to be paid by provider. Using advanced reservation mechanisms enables to guarantee QoS level, by allocating desired amount of capacity, specified by number of nodes and usage-time-period, within the nodes are allocated exclusively. However, the customer has to pay the full price on allocated cpu-hours, independent on the actual usage.

As mentioned in previous section, **scheduling policies** determine (optimal) allocation of resources to jobs according to objective function describing the costs of complete schedule preferably as a single value [32][28]. The simple objective functions are utilization, average response-time, job-throughput etc. However, as optimal scheduling is NP hard problem, approximation algorithms determine scheduling in polynomial time, using heuristics by (1) sorting jobs in the queue at submit time according scheduling-criteria such as job-size, job-length, job-arrival-time, etc., and, (2) putting jobs out of the queue at start time, using first fit or best fit methods. In order to optimize scheduling, in the sense of leaving as less resources unassigned as possible, backfilling strategies such as EASY backfilling or conservative backfilling are used to fill out the gaps in the schedule. The quality of scheduling, expressed as scheduling costs, are measured by performance metrics, such as utilization, average response-time, job-throughput, etc. which influence business metrics, as already motioned.

In conclusion, in this section we presented key-factors and their relationships describing:

1. influence of business-metrics on scheduling behavior, by setting scheduling-objective function, such as utilization, response-time, etc. derived from the business requirements
2. influence of performance metrics on business metrics, by expressing a profit function and their dependency on performance metric and capacity management
3. influence of scheduling objective function on selection of right scheduling policies and scheduling criteria, approximating scheduling objective
4. influence of scheduling policies on performance metrics

In the next section, we identify business policies, affecting key-factors presented in this section.

VII. Business Policies and their Influece on Job-Scheduling

As mentioned in Section IV, for the right configuration of job-scheduling behavior it is essential to understand:

(1) What are the business requirements, expressed by business policies, that influencing job-scheduling?
(2) Where are these requirements/business policies coming from?
(3) How are they influencing job-scheduling?

In this section we investigate these questions, relating them to identified Key-Factors, relationships, and business policies.

A. *Business Policies and Business Metrics*

As stated in earlier work [38], business policies are control statements that guide behavior in a company and control business processes that manage resources (HPC resources, licenses, and people) [37]. The purpose of business policies is to ensure the alignment of business processes with business goals that respond to business requirements [37]. Following OMG's Business Motivation Model (BMM) [37], business policies can define what can be done, what must not be done, and may indicate how, or set limits on how it should be done. Business policies exist to guarantee that the course of action (what has to be done in terms of channeled effort to achieve desired results using resources, skills, competency etc.) will be applied intelligently and within the boundaries of what is acceptable or optimal [37] for the HPC provider. Business policies are not directly enforceable, they require interpretation (e.g. in business rules) and serve as basis for definition of business rules [37]. As noted by Weigand et al. [39] "application of business policies in specific contexts leads to business rules, i.e., highly structured, discrete, atomic statements carefully expressed in terms of a vocabulary to enforce constraints (integrity rules), to deduce new information (derivation rules) or to trigger actions on satisfied conditions (reaction

rules)". Constraints are usually expressed in terms of deontic logic, stating permission/prohibition/obligation/omission, whereas definition rules are expressed typically in form of derivation rules [39]. According to Weiden et al. in [39], business rules can be classified according their semantic properties into: structural, behavioral and managerial rules. Structural and behavioral rules correspond to constraints and definition rules, whereas managerial rules refer to goal-statements. In order to quantify achievement of goals, these are expressed in terms of metered objectives. These objectives are metered (but are not limited) by business metrics, as defined in Section VI.B. For example, managerial rule might be: "The number of violated SLAs for class silver must be lower than 5%". Thereby, "number of violated SLAs" refer to metric for SLA violations in particular QoS class "silver", as noted in Section VI.B, whereas "must be lower than 5 %" prescribes a constraint using obligation ("must"), with comparative operator "lower than" and value of "5%". The corresponding behavioral rule to achieve this objective could demand to increase priority of jobs of the QoS class silver, or to allocate more capacities in advance to silver class.

Hence, business policies are to be considered as a set of highly structured business rules (integrity rules, derivation rules, reaction rules), expressed in terms of vocabulary to be applied in specific contexts to achieve goals quantified by measurements and business metrics, as described in Section VI.B. The following subsections provide examples of business policies, relating to different sources influencing scheduling behavior.

B. Sources of Business Policies and their Influece on Job-Scheduling

As mentioned in earlier work [38], business policies in the context of HPC come from different sources and affect several domains. They might have direct or indirect influences on job-scheduling. Following sources of business policies have been identified to influence job-scheduling:

- Contract Management
- SLA Management
- License Management
- Security Management
- Resource Management
- Accounting Management

The following sub-sections, published in earlier work [38], explain these relationships in detail, showing the scope of business policies that influence job scheduling behavior.

1) Contract Management

Contract Management refers to establishing long-term agreements between a provider and a customer, business partner, (financial) stakeholder, or a third party. Contracts between provider and customer define scope and level of services to be delivered, including agreement on Quality of Services: time (execution time, deadline), costs (rewards, penalties), level of reliability, level of trust/security etc. [8]. Contracts between provider and business partners or stakeholders define constraints, or references to external regulations and policies, that influence scheduling by, e.g., prescribing the usage of HPC resources in a certain way. For instance a contract between a HPC provider and a federal authority that co-financed a HPC cluster could contain regulations prescribing to "use 50 % of the cluster for industrial users and 50 % for researchers for each month". This means that a job scheduler has to limit the CPU time budget for each user group to 50 % of the total CPU time within a period. Another contract between the HPC provider and a federal authority may prescribe to use a HPC cluster in such a way that justifies its huge size. A job scheduler can satisfy such a demand by preferencing large jobs, i.e., using Large Job First scheduling strategy.

Thus, contracting identifies and defines business policies which influence and control job scheduling by setting constraints (i.e. limiting time budget, restricted access to particular resources), by prescribing criteria (job size) and possibly strategy (largest job first) for job scheduling. In addition, contracted QoS between customer and provider define scheduling criteria (i.e. deadline) and constraints that need to be satisfied by scheduling. Business policies in scope of contracting define and constrain the spectrum of HPC service provisioning. They can contain legal statements and references to external regulations and policies.

2) SLA Management

As stated [38], SLA based job-scheduling has been investigated in various fields from different perspectives. In general, SLAs are contracts containing rewards in case of successful execution of job, and penalties in case of violation of QoS, contracted in SLA. SLAs are typically contracted on per-job basis, which means a unique SLA is established for each job to be submitted [38]. SLAs provide more flexibility, enabling provider to offer free capacities in short time-period, thus enabling to increase resource utilization and to fulfill a large number of requests [12][13] [38]. The parameters in SLAs reflect usually job parameters, such as job start and finish times, expected run times, number of requested CPU nodes [40], required processor, time, required disk space etc [38]. However, in case of using SLAs as long term contracts, service levels have to be defined in different way, relating not to particular jobs, but to bundle of jobs or service classes with particular QoS requirements. SLA might then define not only the average response-time for particular job-bundle or service class, but also, i.e., feasible number of violations, such as "the number of violated SLAs for class silver must be lower than 5%". Requirements for particular services classes will force provider to plan carefully available capacities, to reduce potential of SLA violations that affects profit directly. Hence, adherence to QoS levels as defined in SLAs [38] will play major importance for capacity planning and configuration of schedulers, as stated in Section VI.C. "HPC providers nowadays still provide mostly best-effort service without sophisticated QoS levels, but urgent computing, for

example, already calls for a prioritization of customers" [38]. Thus, there is a need for creating different service classes, i.e., "bronze class" for best-effort and "silver class" for jobs which are prioritized [38]. However, this simple distinction on prioritization opens already questions that need to be answered when deciding the scheduling of an individual job which was submitted in reference to an SLA [38]:

- How are jobs in the same service level prioritized against each other?
- If many jobs from a higher service level are being queued, how will jobs from lower service levels be handled?
- How many service levels can be offered (only "bronze" and "silver", or maybe a "gold" service level corresponding to urgent computing)?
- Is profit a key target? In how far is customer satisfaction accounted for?
- How many SLAs can be contracted so that profit and/or customer satisfaction are still satisfying?
- Will lower service levels possibly be starved?

Increasing spectrum of offered service levels (timed access, guaranteed environments, even exclusive access etc.) constrain the provider even more, increasing demand for capacity planning [38], as handled in Section VI. Widening the spectrum of various service levels by offering new service levels, including urgent computing, will potentially attract new customers [38].

3) Accounting

Accounting stores and maintains information about executed jobs, containing number and type of CPUs, duration of the job-execution, and total CPU time spent for the job execution. This information is processed for charging a customer, checking his/her account balance for limits, or for planning future resource allocation decisions [8]. In case of using , i.e., a fairs-hare policies, a job-scheduler might use accounting data to determine total consumed CPU time spent by a user for calculation of his/her jobs in the past, to adjust (increase/decrease) priority of his/her current jobs.

As an example, a business rule might state to "decrease priority of jobs if user has spent more than 95% of his time-budget". Other business rule might state to "allow processing of jobs, only if current accounting balance is greater than 0". Further business rule might state "50% of cpu-time on cluster X must be granted to user-group researchers", and "50% of cpu-time on cluster X must be granted to user-group industrial users". Thus, accounting can be used to check aggregated usage of resources to monitor fair-share, relating to users, projects, customers etc.

4) Security

Security Management is responsible for planning and managing a defined level of security for HPC resources and services. Security policies manage access to HPC resources [8]. They ensure that jobs with requested security level are executed on HPC resources with corresponding security level. For instance, jobs with highly sensitive and confidential information (i.e. crash simulations of a new car model) are executed on HPC resources with high security level. This could be realized by partitioning cluster and allocating resources in dedicated manner for particular jobs, preventing other users from access. A corresponding business rule might demand "jobs of service class gold, must be executed on dedicated partition of the cluster X". Hence security regulate access to HPC resources while meeting requirements of HPC provider and its users.

5) Licencse Management

License Management is responsible for monitoring the availability of licenses and only permits the initiation of a new job if enough licenses are available for its execution. Hence, job schedulers need to take availability of license into account, to create an optimal scheduling. In order to distinguish between different service levels, a corresponding license rule might state to reserve xx licenses of software YZ to service class gold.

6) Resource Management

A resource management system (RMS) is responsible for resource management, job queuing, job scheduling and job execution. Resource management system consists of a resource manager and a job scheduler [13]. Most resource managers have an internal, built-in job scheduler, which can be substantiated by external scheduler with enhanced capabilities, i.e., with support for various scheduling policies like Maui [14]. Resource managers provide schedulers with information about job queues, loads on compute nodes, resource availability etc. Based on that information, a scheduler decides on how and when to allocate resources for job execution. The decision of the scheduler follows a scheduling policy that determines the order in which the competing users' jobs are executed. For example a business rule stating "jobs of industrial users have higher priority than jobs of researchers", would directly influence scheduling by prioritizing corresponding jobs. In addition, a business rule stating "jobs of researchers might be preempted by jobs of industrial users" would lead to immediate preemption of jobs submitted by researchers.

C. Summarizing Influence on Scheduling

As stated [38], License, Security and Resource Management provide the job scheduler with the information on available licenses and resources (quantity and quality), with corresponding security level. In addition, Resource Management provides information on submitted jobs waiting in the job queue. The identified key-factors, as presented in Section VI.C, are to be considered as the

information on job requirements and available capacities (resources, supported security level, licenses).

Accounting Management provides the job scheduler with information on job submission and resource consumption history, indicating used resources and consumed CPU time per user, user group, or project. Identified key factors (data-history and remaining budget) are to be considered as information to the job scheduler, allowing to control fair-share policies, identifying and predicting workload behavior of users, and adapting scheduling behavior to achieve the best possible scheduling performance or fairness between users.

In scope of contracting identified Stakeholder Management comprise contracts between the HPC provider and its stakeholders. These contracts, containing legal statements, define boundary conditions on job scheduling. Identified key factors regulate the usage on HPC resources on high level, constraining directly or indirectly (by deriving from the legal statements) high level scheduling behavior by characterizing a range of possible scheduling criteria. The definition of the utility function (for the provider), calculating utility and benefit for each job, must take these constraints into account, determining range of permissible scheduling criteria and objective functions. In case of conflicting constraints of different stakeholders, conflict resolution strategies are required to resolve conflicts, i.e., according to validity of constraint or importance/prioritization of stakeholder.

Customer Relationship Management (CRM) comprises SLA Management and Contract Management. Identified key factors from these domains involve QoS parameters, such as time (job wait time, latest job deadline, estimated job run time, etc.), type and amount of required resources, requested security level, and costs (rewards and penalties). In addition to these key factors, which are reflected in SLAs and contracts, there is a key factor called ”importance” which expresses how important a customer or a project is for the provider. The ”importance” key factor expresses the preferences between different customers, and might be based on contracted service level, rewards/penalties, strategically long term partnership, and on other subjective criteria. These key factors are to be considered as parameters for the job scheduling policies. A job scheduling policy can be defined by selecting one or combining several of these key factors into scheduling criteria. The question, in how far the scheduling behavior must adhere to QoS (as contracted in SLAs or contracts), is outside of the CRM view, as well as the control (in the sense of the definition of the scheduling function) on the job scheduling behavior. Contracted rewards and penalties provide only a financial assessment on fulfillment or violation of SLAs/contracts, they don’t form 100 % guarantees. On the other hand, service providers exist on basis of quality of the customer services. Hence the provider should not violate SLAs/contracts, if possible. However, in case of overloading situations, where existing HPC resources are not sufficient to fulfill all SLAs and contracts, the provider has to make prioritization between customer’s jobs, which can be based on ”job deadline” (urgency), ”customer importance” and on other business objectives of the provider.

The business objectives of the HPC provider are determined by its mission and vision statements. Dependent whether the provider is profit-oriented or not, there are different business objectives. A profit-oriented HPC provider, could define the high level objective as ”maximum profit”. Hence, the job scheduling strategy would be to maximize the overall sum of rewards, obtained by fulfilling SLAs. A possible scheduling policy would be to sort jobs in the queue according to their deadlines (earliest deadline first) and rewards (maximum reward). A non-profit oriented HPC provider (a university, for example) has typically a mission to ”promote research” by providing students and researchers access to HPC resources. As researchers’ jobs are equally important, the overall goal of the provider is to ”maximize number of completed jobs”. A possible scheduling policy would be FCFS with FF (First Fit) strategy, which ensures fairness and increases the number of jobs completed.

VIII. Using SBVR for Definition of Business Policies

In this section we introduce Semantics of Business Vocabulary and Business Rules (SBVR) intended to describe business policies. We provide examples describing business policies, consistence checking rules and transformational rules, allowing to translate business policies into scheduling policies.

A. *Semantics of Business Vocabulary and Business Rules*

As stated, OMG’s Semantics of Business Vocabulary and Business Rules (SBVR) [31] in its version 1.0, is recent (2008) standard intended to define “the vocabulary and rules for documenting the semantics of business vocabularies, business facts, and business rules” [31], serving as basis for the natural language declarative description of complex entities and rules. In SBVR, business facts and business rules may be expressed either informally or formally, capable to be interpreted and used by humans and computer systems [31]. Formal statements are “expressed purely in terms of fact types (verb concept) in the pre-declared schema of the business domain, as well as certain logical and mathematical operators, including quantifiers” [31]. Terms or vocabularies in SBVR are used to describe the formal semantic structures of discourse domain, using semantic formulations based on logical composition of meaning [31]. Only formal statements may be transformed to logical representation in first order predicate logic with a small extension in modal logic, enabling consistency checking between rules.

SBVR follows business rule mantra, where “Rules are based on facts, and facts are based on terms” [31]. Thereby, “a fact is a proposition taken to be true by the business” [31], and serves as a basis of communication [39].

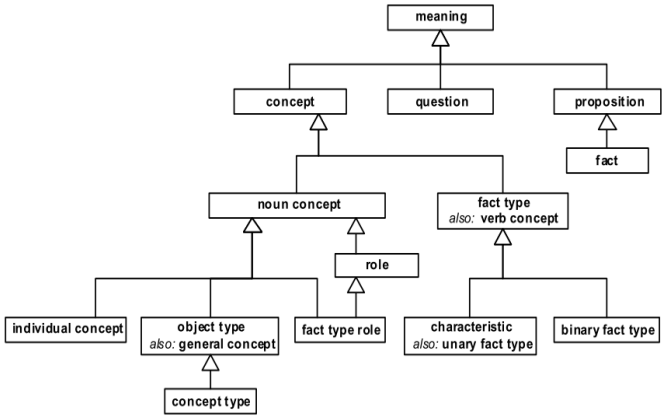

Figure 3. SBVR Metamodel and Vocabulary [31].

Terms and vocabularies (concepts) in SBVR, as shown in Figure 3, are defined by noun concepts and fact-types (or verb concept). Noun concepts express individual concepts (instance of a concept that corresponds to only one object), object-types (general concept class), and roles (concepts that correspond to things based on their role). Noun concepts form class hierarchies via subtype relationships, such as specialization and generalization providing the basis for subsumption reasoning [41]. Fact types (also called verb concepts) describe relationships among concepts, including unary (describing characteristic of a concept), binary (relationship between two concepts) and n-ary (relationships among roles, with fixed number of roles) relationships [41]. Attributive fact types capture mereological relationships, such as relationships between parts and a whole [41]. For example, a rule stating "Scheduling policy *has* scheduling criteria" defines fact-type, describing that every scheduling policy has a property called scheduling criteria. Scheduling criteria can be defined as a value used for sorting of jobs according to particular job-characteristics.

Rule statements in SBVR, as shown in Figure 4, can be divided into Structural Business Rules and Operative Business Rules. Structural rules are definitional rules, proposing necessary characteristics of concepts or models, being always true for each instance of the concept. Structural rule statements often facilitate a deeper understanding of concepts, but a structural rule never changes a concept [31]. Structural rules use two alethic modalities, expressing logical necessity ("it is necessary that…") and logical possibility ("it is possible that…"). For example, "**It is possible that** a scheduling policy *has* **more than one** scheduling criteria", expresses that several criteria might be used for job-scheduling. Behavioral rules describe guidance specifying expectations that can be violated by people or systems by not following them [41]. Behavioral rules are described using deontic modalities, expressing obligation ("It is obligatory that …") and permissions ("It is permitted that…"). For example, "**It is obligatory that** jobs of gold customers are started within 5 hours".

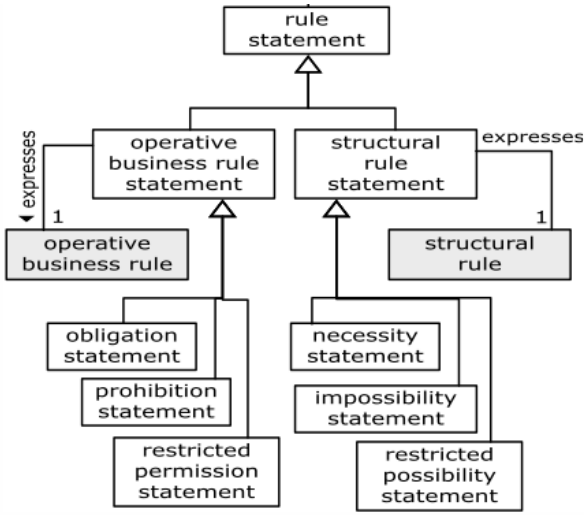

Figure 4. Rule statements in SBVR [31].

The examples of SBVR rules in subsequent sections are given in "Structured English", using format and font styles as suggested by Linehan [41]:

nouns are underlined
verbs are given in italics
literal values and instance names are double underlined
keywords are shown in bold font
uninterpreted text is shown in normal font style

B. *Using SBVR to describe Business Policies for Job-Scheduling*

The proposed approach to "business policy based resource-management/job-scheduling in HPC" uses SBVR for the description of business vocabularies, facts, and rules, to ensure compliance between business policies and scheduling policies. Following sections provide simple examples for definition of vocabularies and rules, which are used for the transformation and consistency checking between business objectives, scheduling objectives and scheduling policies.

1) *Defining Vocabulary*

As mentioned before, vocabularies and terms in SBVR are defined using noun concepts and fact types. Firstly, we start with the definition of concepts that are central for scheduling, describing concepts and relationships between "scheduler", "scheduling objective", "scheduling policy" and corresponding characteristics of particular "scheduling policies" as instances of fact types:

Scheduler *has* scheduling policy	(R1)
Scheduling policy *has* scheduling criteria	(R2)
Scheduling policy *has* performance indicators	(R3)
Utilization *is a* performance indicator	(R4)
Response-time *is a* performance indicator	(R5)
Scheduling policy *has* scheduling objective function	(R6)

minimize response time *is a* scheduling objective function (R7)

maximize utilization *is a* scheduling objective function (R8)

2) Expressing Scheduling Policies

In the next step, we define scheduling policies, explaining their meaning based on their kind of sorting of jobs (up or down) according to specific scheduling criteria. For this purpose we define a corresponding fact type as follow:

A Scheduling Policy *sort* (up or down) jobs in the waiting queue according to **specific** scheduling criteria

For simplicity, we reformulate it as follow:

A Scheduling Policy *sort* (up or down) according to **specific** scheduling criteria (R9)

Consequently, the definition of scheduling policies is to be considered as instances of previously defined fact type:

LJF *is a* scheduling policy that *sort* down according to scheduling criteria job-length. (R10)

SJF *is a* scheduling policy that *sort* up according to scheduling criteria job-length. (R11)

LSJF *is a* scheduling policy that *sorts* down according to scheduling criteria job-size. (R12)

SSJF *is a* scheduling policy that *sorts* up according to scheduling criteria job-size. (R13)

In order to enable detection of inconsistence and contradictions between various policies, we define up as opposites of down, and vice verse:

up *is not* down (R14)

down *is not* up (R15)

3) From Scheduling Policies to Scheduling Objectives

As mentioned before, scheduling policies are approximations of the scheduling objective functions. For example, LJF and LSJF are greedy strategies aiming at maximizing utilization, as they acquire as long/much resources as possible, according to maximum job-length/job-size. In contrast, SJF and SSJF are greedy strategies aiming at minimizing average response-time, as they acquire as short/little resources to jobs as possible, according to minimum job-length/job-size.

We can specify simplified (and idealized) rules that describe these relationships in different way, as transformation rule and as a compliance rule. We start with the definition of compliance rules. As there are many scheduling policies aiming at, i.e., maximizing utilization, we can define a compliance rule that checks whether the actual scheduling policy is compliant with current scheduling objective:

Scheduling policy that *sort* down for *any* scheduling criteria, *has* objective function *maximize* utilization (R16)

Scheduling policy that *sort* up *for any* scheduling criteria, *has* objective function *minimize* response-time (R17)

A transformational rule defines how to translate from objective function to scheduling policy. As there are different degrees of enforcement or advice existing (permission, obligation,…), we define a transformation rule as an permission, allowing selection of several alternatives - corresponding to "1 to n" mapping (from scheduling objective to scheduling policy):

It is permitted that scheduling objective function that *maximize* utilization *may* use scheduling policy that *sort* down *for any* scheduling criteria. (R18)

It is permitted that scheduling objective function that *minimize* response-time *may use* scheduling policy that *sort* up *for any* scheduling criteria. (R19)

Alternatively it can be reformulated as follow:

Administrator *may use* scheduling policy that *sort* up for *any* scheduling criteria, **only if** scheduling objective function *is minimize* response-time (R20)

Administrator *may use* scheduling policy that *sort* down for *any* scheduling criteria, **only if** scheduling objective function *is maximize* utilization (R21)

Using deontic equivalence rules, these rules can be reformulated as:

Administrator *must not use* scheduling policy that *sort* down for *any* scheduling criteria, **if** scheduling objective function *is not maximize* utilization

Correspondingly:

Administrator *must not use* scheduling policy that *sort* up for *any* scheduling criteria, **if** scheduling objective function *is not minimize* response-time

A "permission" (with may statement) expresses optional selection of particular action for particular condition, whereas "prohibition" with "must not" statement prohibits the selection of particular option/action on inverted condition. It should be noted that in case of using "obligation" instead of "permission", it restricts the transformation space to "1 to 1" mapping. However, in general case, "permission" should be used instead of "obligation", to allow selection of several alternative scheduling polices, thus enabling "1 to n" mapping.

4) From Scheduling Objectives to Business Goals

As described in VI.B, relationship between profit function and utilization can be described in the following way:

Profit *is a* difference *between* revenue and costs (R22)

Profit = revenue – costs (– penalty) (R22)

Revenue *is* a product of utilization and price (R23)

Revenue = utilization* price (R23)

With fixed price:

Price *is* fixed (R24)

and fixed costs:

Cost *is* fixed (R25)

Thereby we assume, semantic of functions, such as sum, difference, product, maximization and minimization are defined using mathematical functions. Alternatively,

maximize can be defined, dependent on variable and fixed parameters, semantically in the following way:

If *maximize* difference of variable X and fixed Y **then** *maximize X* (R26)

If *maximize* product of variable X and fixed Y **then** *maximize X* (R27)

To allow transformation between fact-type and its nominalization, we define following rules:

maximize utilization *is* maximize utilization (R28)

maximize utilization *is* scheduling objective function (R29)

Before we start with the definition of business goals, we need to define terms that are used for description of these business goals. For example, "maximizing profit" as a business goal can be expressed as a fact type that *maximizes* profit:

Maximum profit *is a* business goal that *maximize* profit (R30)

To indicate which/what goal should be pursued by HPC provider, we define operational rule, prescribing obligatory to use "maximum profit" as a business goal:

It is obligatory *to use* business goal maximum profit (R31)

In the next section we describe the full cycle of transformation and consistency checking, using pre-defined rules.

5) From Busines-Goal to Scheduling Policies

Using previously defined rules allows transformation from business goal to scheduling policies, allowing at the same time checking consistency between rules. It is clearly to see, that following the business goal "Maximum Profit" implies:

(R31)→ Maximum profit

(R30) → *maximize* profit

(R22), (R25), (R26) → *maximize* revenue

(R23), (R24), (R27) → *maximize* utilization

(R28), (R29) → scheduling objective function maximize utilization

(R18) → scheduling policy that *sort* down *for any* scheduling criteria.

(R10) → use LJF

or

(R12)→ use LSJF

Thus, LJF or LSJF scheduling policies can be used for configuration of job-scheduler, to achieve "maximum profit".

However, as new scheduling policies might be defined using various scheduling criteria, it is necessary to analyze these policies to identify corresponding scheduling objective function, approximated by scheduling policy. A possible approach allowing identifying relationship between scheduling policies and scheduling objective function can be based on greedy heuristics, as explained previously. Other heuristics might be used as well.

SBVR approach presented in this section allows by definition of vocabularies and rules, transformation of business goals and business policies into scheduling policies. Nonetheless, as a transformation is "1 to n" mapping (from business policies to scheduling policies) where several scheduling policies are possible for the same goal, and selection of the right scheduling policy might depend on additional job-characteristics, such as job-submission-time, job-arrival rate, variance on job-size and job-length, etc., the transformation rules may be refined using additional criteria or experience made in the past. The definition of policies described in VII can be defined in a similar way, starting with the definition of terms and concepts used in these business policies, continuing with definition of fact-types – serving as schema, and resulting in definition of business policies and rules, prescribing a goal, a constraint or a behavior (condition – action rule).

However, examples presented in this section covered only idealized basic approach, not considering internal / external influences from various sources, as presented in VI.B. These aspects will be covered in future work, enabling evaluation of various (internal/external) influencers of business policies and goals.

IX. SUMARY AND FUTURE WORK

In this paper we outlined why business policy based jobs-scheduling is needed, presenting an approach allowing to investigate how much influence business policies have on job-scheduling in HPC domain. The proposed bottom-up process explains identification of relationships between scheduling policies and business policies in several steps, including scheduling-performance-indicators, and key-factors. Following this approach we analyzed in Section VI.A scheduling policies, indentifying scheduling criteria used by scheduling policies, scheduling objective function approximated by scheduling policies, and performance metrics/indicators characterizing costs of scheduling. In Section VI.B, we identified relationships between performance metrics and business metrics, providing corresponding definition of business metrics. The results of the first two steps were summarized in Section VI.C into a model, described by key-factors and relationships influencing scheduling behavior. In Section VII, we described influence of business policies on business metrics and scheduling behavior, considering various sources of influence. Finally, in Section VIII, we described usage of SBVR as business policy language, for definition of business vocabularies, business rules, facts and business policies related to job-scheduling in HPC domain, capable to check consistency between rules or to describe transformation between business policies and scheduling polices. The

general aim of the proposed approach to realize hierarchical policy refinement, allowing transformation of business policies together with other constraints into selection and configuration of parameters and policies needed to configure policy based schedulers was demonstrated on few examples presented in Section VIII.B.

Results presented in this paper described theoretical basis of the proposed approach on "business policy based resource-management/job-scheduling in HPC", and require implementation of all policies, rules and terms, to be covered in future work. Future work will also comprise evaluation of policies on various levels, including business-policies, business-metrics, scheduling-performance-metrics and scheduling criteria, to detect inconsistencies and conflicts between business policies. In the next step, detected conflicts will be assessed according to their influence on resource-usage and business-impact. The purpose of resulting tool is to support business people in design and definition of business policies, allowing assessing impact of varying business policies and possible conflicts on service provisioning in HPC.

ACKNOWLEDGMENT

The results presented in this paper are partially funded by Federal Ministry of Education and Research (BMBF) through the TIMaCS project [17].

REFERENCES

[1] Volk, E.: Approach to Business-Policy based Job-Scheduling in HPC, Cloud Computing 2010, Lisbon, November 2010, pp. 20-25.

[2] Maui Scheduler Administrator's Guide, version 3.2 from http://www.adaptivecomputing.com/resources/docs/maui/, [Last access January 23, 2012]

[3] Moab Workload Manager Website. http://www.adaptivecomputing.com/products/moab-hpc.php, [Last access January 23, 2012]

[4] Cron Wikipedia description, from http://en.wikipedia.org/wiki/Cron, access date 17.06.2010

[5] IBM, "Policies and Rules improving business agility", IBM website, http://www.ibm.com/developerworks/webservices/library/ws-policyandrules/index.html, [Last access January 23, 2012]

[6] S. Iqbal, R. Gupta, Y. Fang, *Planning Considerations for Job Scheduling in HPC Clusters*. Dell PowerSolutions, Feb 2005

[7] T. L. Casavant, G. J. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems". *IEEE Transactions on Software Engineering* 1988; 14(2):141–154.

[8] C. S. Yeo, R. Buyya, "A taxonomy of market-based resource management systems for utility-driven cluster computing." in *Software-practice and experiences* 2006; 36:1381–1419, Published online 8 June 2006 in Wiley InterScience

[9] J. H. Abawajy, "An efficient adaptive scheduling policy for high-performance computing", in *Future Generation Computer Systems*, Volume 25, Issue 3, March 2009, Pages 364-370.

[10] R. Boutaba, I. Aib, "Policy-based Management: A Historical Perspective", *Journal of Network and Systems Management*, pp 447-480, Springer, 2007

[11] IBM, "An architectural blueprint for autonomic computing.", *IBM Whitepaper*, June 2005, http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf, [Last access January 23, 2012]

[12] R. Sakellarioiu, V. Yarmolenko, "Job Scheduling on the Grid: Towards SLA-Based Scheduling." in *High Performance Computing and Grids in Action*, pages 207–222. IOS, 2008.

[13] V. Yarmolenko, R. Sakellariou, "An Evaluation of Heuristics for SLA Based Parallel Job Scheduling." *3rd High Performance Grid Computing Workshop* (in conjunction with IPDPS 2006), 2006.

[14] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, R. Buyya, "Libra: Economy-Driven Job Scheduling System for Clusters.", in *Software: Practice and Experience* 2004; 34(6):573–590.

[15] L. Tang, Z. Yang, Z. Yu, Y. Wang, "A Quality-Driven Algorithm for Resource Scheduling Based on Market Model on Grid.", 2007 International Conference on Parallel Processing Workshops (ICPPW 2007)

[16] M. Hondo, J. Boyer, A. Ritchie, "Policies and Rules – Improving business agility: Part 1: Support for business agility", IBM Whitepaper, 16. March 2010

[17] TIMaCS - Tools for Intelligent Management for Very Large Computing Systems, Web site: www.timacs.de, [Last access January 23, 2012]

[18] Scheduling Wikipedia description, from http://en.wikipedia.org/wiki/Scheduling_(computing), [Last access January 23, 2012]

[19] W. T. Greenwood, "Business Policy-Case Method Forum: A Rejoinder", in *The Academy of Management Journal*, Vol. 10, No. 2 (Jun., 1967), pp. 199-204

[20] C. Kandagatla, Survey and Taxonomy of Grid Resource Management Systems, University of Texas, Austin. [Online] Available:http://www.cs.utexas.edu/users/browne/cs395f2003/projects/KandagatlaReport.pdf.

[21] A. Moura, J. Sauve, C Bartolini „Research Challenges fo Business-Driven IT Management", in Business-Driven IT Management, 2007. BDIM '07. 2nd IEEE/IFIP International Workshop on BDIM

[22] J. Oriol Fito and J. Guitart, "Initial Thoughts on Business-driven IT Management Challenges in Cloud Computing Providers", 6th IFIP/IEEE International Workshop on Business Driven IT Management, 2010

[23] J. P. Sauvé, J A. Moura, M. C. Sampaio, J. Jornada, E. Radziuk, "An Introductory Overview And Survey Of Business Driven It Management", 1st IEEE / IFIP International Workshop On Business-Driven Management, IT Management

[24] J. P. Sauvé, R.R. Almeida, J A. Moura,J. A. Belträo, C. Bartolini, A. Boulmakoul, D. Trastour, "Business-Driven Decision Support for Change Management: Planning and Scheduling of Changes." In: 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2006, 2006, Dublin, Irlanda. Large Scale Management of Distributed Systems. Heidelberg : Springer Berlin, 2006. v. 4269. p. 173-184.

[25] D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, and K. C. Sevcik, "Theory and Practice in Parallel Job Scheduling." In D. G. Feitelson and L. Rudolph, editor, Proc. of 3rd Workshop on Job Scheduling Strategies for Parallel Processing, volume 1291 of Lecture Notes in Computer Science, pages 1–34. Springer Verlag, 1997.

[26] D. G. Feitelson and L. Rudolph. "Metrics and Benchmarking for Parallel Job Scheduling." In D. G. Feitelson and L. Rudolph, editor, Proc. of 4th Workshop on Job Scheduling Strategies for Parallel Processing, volume 1459, pages 1–24. Springer Verlag, 1998.

[27] A. Streit, "On Job Scheduling in HPC-Clusters and the and the dynP Scheduler", In High Performance Computing — HiPC 2001, Vol. 2228 (4 December 2001), pp. 58-67.

[28] A. Streit, "Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids", Dissertation, 2003, online http://digital.ub.uni-paderborn.de/ubpb/urn/urn:nbn:de:hbz:466-20030101378, [Last access January 23, 2012]

[29] M. Hovestadt, O. Kao, A. Keller, A. Streit, "Scheduling in HPC Resource Management Systems: Queuing vs. Planning". In Job

Scheduling Strategies for Parallel Processing, Vol. 2862 (2003), pp. 1-20.

[30] D. G. Feitelson. “A Survey of Scheduling in Multiprogrammed Parallel Systems.” Research, report rc 19790 (87657), IBM T.J. Watson Research Center, Yorktown Heights, NY, 1995.

[31] OMG group, “Semantics of Business Vocabulary and Business Rules” (SBVR), version 1, January 2008, online http://www.omg.org/spec/SBVR/1.0/, [Last access January 23, 2012].

[32] J. Krallmann, U. Schwiegelshohn, and R. Yahyapour. On the Design and Evaluation of Job Scheduling Algorithms. In D. G. Feitelson and L. Rudolph, editor, Proc. of 5th Workshop on Job Scheduling Strategies for Parallel Processing, volume 1659 of Lectures Notes in Computer Science, pp. 17–42. Springer, 1999.

[33] S. K. Garg, C. S. Yeo, A. Anandasivam, R. Buyya, "Energy-Efficient Scheduling of HPC Applications in Cloud Computing Environments", 2009, url: http://www.cloudbus.org/reports/EE-SchedulingAcrossClouds-2009.pdf, [Last access January 23, 2012]

[34] B. Abrahao, V. Almeida, J. Almeida, A. Zhang, D. Beyer F. Safai, "Self-Adaptive SLA-Driven Capacity Management for Internet Services", 2006, in 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM.

[35] M. Siddiqui, A. Villazion, T. Fahringer, "Grid capacity planning with negotiation-based advance reservation for optimized QoS", in SC'06 Proceedings of the 2006 ACM/IEEE conference on Supercomputing, 2006

[36] S. Tichenor, A. Reuther: "Making the Business Case for High Performance Computing: A Benefit-Cost Analysis Methodology", in CTWatch Quarterly, Volume 2 Number 4a, November 2006

[37] OMG: Business Motivation Model, Version 1.1, Release May 2010, http://www.omg.org/spec/BMM/1.1/, [Last access January 23, 2012]

[38] E. Volk, R. Kübert (2010) "Towards Business-Policy based Job-Scheduling in HPC", pp 126 - 135, in Proceedings of Cracow Grid Workshop 2010.

[39] H. Weigand, W.J. van den Heuvel, M. Hiel, "Business policy compliance in service-oriented systems", 2011, in Information Systems, v.36 n.4, p.791-807, June, 2011

[40] C. S. Yeo, R. Buyya: "Managing Risk of Inaccurate Runtime Estimates for Deadline Constrained Job Admission Control in Clusters." In Proceedings of the 2006 International Conference on Parallel Processing (ICPP ’06). IEEE Computer Society, Washington, DC, USA, 451-458. DOI=10.1109/ICPP.2006.52 http://dx.doi.org/10.1109/ICPP.2006.52, [Last access January 23, 2012]

[41] M. H. Linehan: “SBVR Use Cases”, Advancement of Artificial Intelligence (AAAI), 2008