# A Decision Maxim for Efficient Task Realization within Analytical Network Infrastructures

M. Grum[a,*], B. Bender[a], A.S. Alfa[b,c], N. Gronau[a]

[a] *Department of Business Informatics, especially Processes and Systems, University of Potsdam, Potsdam, Germany*
[b] *Department of Electrical, Electronic and Computer Engineering,University of Pretoria, Pretoria, Gauteng, South Africa*
[c] *Department of Electrical and Computer Engineering University of Manitoba, Winnipeg, Manitoba, Canada*

**Abstract**

Faced with the increasing needs of companies, optimal dimensioning of IT hardware is becoming challenging for decision makers. In terms of analytical infrastructures, a highly evolutionary environment causes volatile, time-dependent workloads in its components, and intelligent, flexible task distribution between local systems and cloud services is attractive. With the aim of developing a flexible and efficient design for analytical infrastructures, this paper proposes a flexible architecture model, which allocates tasks following a machine-specific decision heuristic. A simulation benchmarks this system with existing strategies and identifies the new decision maxim as superior in a first scenario-based simulation.

*Keywords:* Analytics, Architecture Concepts, Cyber-Physical Systems, Internet of Things, Task Realization Strategies, Simulation

## 1. Introduction

Faced with an increase in the complexity of company IT infrastructures, such as an increasing number of networked machines and their heterogeneity in hardware and software (Polyvyanyy et al. (2017)), companies are often chal-

*Corresponding author at.
*Email address:* mgrum@lswi.de (M. Grum)

lenged regarding capacities for the processing of time-critical analytical tasks. Furthermore, tasks must be realized at the lowest possible price, selected customer focus, and desired flexibility, among others. In this case, a trade-off must be determined among various and conflicting criteria. A variety of task distribution approaches within networked infrastructures, each exhibiting characteristic advantages and disadvantages, complicates the distribution and processing of analytical tasks. Prominent approaches include the following: edge computing, which focuses on decentralized processing at the margin of a network, close to the data-generation location; cloud computing, which subsumes the concepts of central task processing by using more powerful but additionally paid resources via the Internet; and fog computing, which focuses on near-user edge devices. (Lopez et al. (2015)). As an increasing number of devices are connected via the Internet, external parties can easily be integrated by means of software as a service (SaaS) concept, so that analytical tasks can be realized in parts, on behalf of external and internal devices. In light of the complex price models of external parties, task distribution becomes further complicated (May Al-Roomi & Ahmad (2013)).

Faced with digitization and dissemination of the Internet of things, physical objects are often enhanced by cyber-physical systems (Vogel-Heuser et al. (2009)). With this, common production settings are enriched, and function as cyber-physical production systems. Here, the variety of processing tasks as well as the amount of data to be processed increase constantly. Tasks that are closely related to the physical value-adding production process demand special requirements regarding real-time task processing and require systems to decide individually (Kopetz (2011)). In this context, the following research questions are pertinent:

1. How can task realization approaches in real-world settings be compared?
2. How can analytical tasks be processed efficiently within networked infrastructures?

Building on the design science research methodology (DSRM) of Peffers et al. (2007), this paper is structured by the *Publication Schema* of Gregor & Hevner (2013). Hence, the following section demonstrates the required theoretical foundation with the aid of basic concepts and identifies the research gap. The third section provides the methodological approach. The fourth section designs the required artifacts. Hereunder, one can find the mathematical model and new decision maxim (NDM), and their computational implementation. Simulation results are provided as a demonstration and evaluated by a performance evaluation framework in the $5^{th}$ section. This illustrates the functioning of the flexible architecture concept. Finally, conclusions are provided in the sixth section.

## 2. Theoretical Foundation

Based on the DSRM, this section provides related concepts required for gaining improved understanding and the solution design. In addition to the concepts, a literature overview provides related work and demonstrates the need for the development of a new decision maxim for efficient task processing within network infrastructures.

### *2.1. Underlying Concepts*

In this section, relevant concepts within the context of business analytics infrastructures are identified. Typical IT infrastructure levels are differentiated as typically found in enterprise setups.

#### *2.1.1. Computing Infrastructures*

Although individual company infrastructures may vary, common processing infrastructure patterns and levels should be used as a basis for optimization. Following Grozev & Buyya (2014), three typical computing infrastructure levels can be differentiated in modern enterprises. In general, computing infrastructures consist of computing systems that can be grouped into the CPS, local cloud or public cloud level.

*CPS level.* This level is at the very bottom and refers to the *shop floor level*. According to Gronau et al. (2016), the computing resources located here subsume different machines and components as variations of cyber-physical systems (CPSs) that are part of the value creation process of a company. Hence, their concrete hardware configurations vary significantly. Using a manufacturing company as an example, the CPS level includes production robots, which participate directly in value creation processes and deal with material flows, and quality control components, which participate indirectly in value creation processes.

*Local cloud level.* Above the CPS level, a level known as the *local cloud* can be found. This architectural level subsumes the more centrally located processing components of an enterprise, which are typically more powerful than a single CPS (Grozev & Buyya (2014)). Its components are commonly interconnected based on local area network and virtual private network technologies, and include different locations, being part of an *intranet* infrastructure (Donahue (2011)). As components are property of the enterprise, they are run and maintained by its IT departments. From an analytical perspective, this typically includes data warehouses for storing relevant information centrally as a basis for downstream systems. Hereunder, business intelligence and reporting software, including dashboards and similar applications, can be found.

*Public cloud level.* The top architectural level is known as the *public cloud*, which is not part of the company infrastructure. It is maintained by third-party providers and utilized by the company for computing tasks. Computing infrastructures on the public cloud level are typically rented from cloud hosting providers that offer computing resources on demand. In general, such infrastructures exhibit high scalability, which provides them with the role as a perfect supplement to the company infrastructure components (Jadeja & Modi (2012)). If tasks cannot be computed effectively within the company infrastructure; for example, because of a small remaining time, high task requirements or computing efforts, they can be computed on the public cloud level, provided that data

privacy and security, as well as necessary constraints (e.g., latency), are ensured (Ren et al. (2012)).

*Relationships among levels.* The processing of analytical tasks can be carried out for components of any of the three levels. Taking various individual resource characteristics into account, Fig. 1 (a) visualizes the three levels and their relationships regarding their *computational power*. In general, a higher analytical architecture level will yield a higher level of computing power. Therefore, higher levels are more suited to heavy, analytical tasks. Furthermore, with increasing levels, the local proximity to the value creation process declines, which leads to additional hardware costs and transfer times.

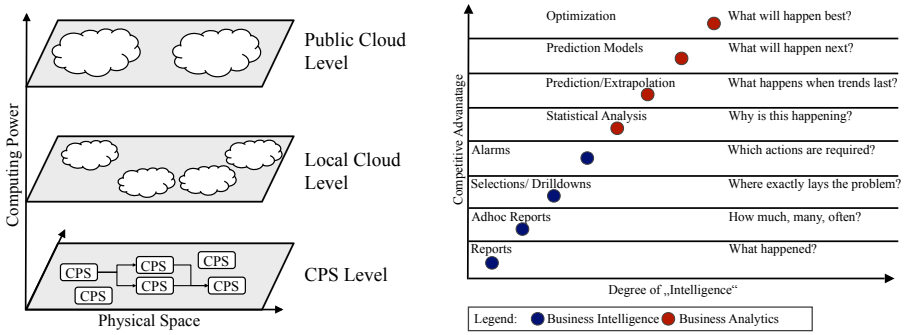
The question as to on which level provides different computing power and costs of analytical tasks shall be realized remains. Hence, the three identified levels serve as a reference point for the design of first simulations.

### *2.1.2. Analytics*

EMC-Education-Services (2015) refer to analytics as a group of analytical techniques dealing with analytical tasks. A first group, known as business intelligence (BI), tends to explain the current or past business behavior by aggregating and grouping mostly well-structured historical data. A second group, known as business analytics (BA), tends to explore the future based on the present and a forward-looking, decision-enabling system. This is generally based on less structured data emanating from various systems.

*Task Types.* According Davenport & Harris (2007), analytical tasks can be mapped to eight task types, as illustrated in Fig. 1 (b). Analytical techniques consider BI (tasks are visualized by blue circles) and BA (red circles), while a demonstration considers both. While the name of each task type is placed next to its circle, corresponding questions can be found in the middle and a symbolic picture of each advantage level on the right. Following the assumption of Davenport and Harris: a higher degree of intelligence for the technique required by a task type results in a higher competitive advantage that can be realized

because of dealing with this task type. Consequently, the effort also increases and a trade-off between the gained advantage and required effort must be determined by each company individually. The task types serve as framework for the analytical tasks within the designs and demonstrations of this contribution.




(a) Analytic infrastructure (b) Available task types

Figure 1: Abstract model of analytical architectures

*Parallelization.* Each task type can be parallelized, depending on the parallelization characteristics of the analytical task. In this case, massively parallel processing architectures enable parallel data ingestion and analysis. Although these are the preferred approach to processing complex data (Wong et al. (2013)), it is not clear from a practical point of view which instance within the analytical infrastructure is best suited best to the processing of a current task. Therefore, the presented dynamic task realization approach must distinguish among the distributions of tasks within networking structures, the processing order in every system, and the corresponding task type.

### *2.2. Related Literature*

In addition to the previously presented concepts, the following research works are related to the topic addressed in this contribution. Table 1 displays their categorizations according to the different domains of infrastructure components, task processing, and dynamics. A dynamic refers to aspects owing to the situational adequate computation of tasks and environmental changes regarding

infrastructure. Considering the variety of related research aspects, their coverage is exemplary.

Table 1: Categorization of related work.

| Contributions | Hardware components | Infrastructure characteristics | Task type requirements | Algorithmic aspects | Situational variations | Environmental changes |
|---|---|---|---|---|---|---|
| Augonnet et al. (2010) | | ✓ | | ✓ | | |
| Bender & Grum (2016) | ✓ | ✓ | | ✓ | | |
| Brooks et al. (2000) | ✓ | | | ✓ | | |
| Cevher et al. (2014) | | ✓ | ✓ | ✓ | | |
| Davenport & Harris (2007) | | | ✓ | | | |
| Gupta & Chow (2010) | | ✓ | | ✓ | | |
| Grozev & Buyya (2014) | ✓ | | | ✓ | | |
| Grum et al. (2017) | ✓ | | | | ✓ | ✓ |
| Pike et al. (2009) | ✓ | | | ✓ | | |
| Polyvyanyy et al. (2017) | | ✓ | | | | |
| Wong et al. (2013) | | | | ✓ | | |
| Zikopoulos & Eaton (2011) | | | ✓ | ✓ | | |

On the level of single computing entities, corresponding to the CPS level, various designs and implementations for efficient computing of several tasks can be determined. Different algorithm types are optimized with regard to various demands and hardware architectures (Cevher et al. (2014)). The research also discusses the design of hardware architectures and its components (Brooks et al. (2000)). Furthermore, mechanisms for scheduling tasks on single computing systems have been developed (Augonnet et al. (2010)).

On the level of networked infrastructures, corresponding to the local and public cloud levels, the coordination and scheduling of distributed task processing has been researched (Pike et al. (2009)). Specialized concepts have evolved for conducting computation tasks in networked environments and related requirements. Examples can be found in shared memory concepts, distributed file systems, and computation concepts, such as Hadoop (Zikopoulos & Eaton (2011)). Therefore, hardware dimensioning has become important. Distributed computing strategies under constraints such as slow network connections have also been developed (Gupta & Chow (2010)). As public clouds in particular incorporate large computing infrastructures, the aspect of energy-efficient computing has become a significant theme (Luo et al. (2012)). While each research study focuses on a certain problem, in accordance with Grum et al. (2017), each focuses on a highly specific optimization problem and from a general, system-wide perspective, only local optima can be identified with each.

Research regarding the complex distribution of analytical tasks, within a given dynamic infrastructure and across different levels, is rare. Previously presented architectural infrastructure levels were identified by Grozev & Buyya (2014), and combined with task realization strategies on the basis of heuristics in analytic infrastructures by Bender & Grum (2016). An integration of available concepts was researched by Grum et al. (2017), with the aim of identifying global optima in analytical task processing over scenarios and environmental changes. A quantified benchmark of different task realization strategies, such as real-world live tests or simulations, remains to be developed.

*Research gap.* Although single aspects of efficient analytical infrastructures within modern infrastructures have been thoroughly researched, a combination of these and benchmark within one common framework is lacking. Each concept is valid within its research domain, but only several studies have considered individual infrastructure characteristics and dynamics regarding tasks and infrastructure changes. None of the identified studies has combined all aspects.

The major contribution of this work is therefore the provision of a flexible architectural framework allowing for the integration of domain-specific approaches, overcoming disregarded infrastructure, tasks, and algorithms as well as dynamics, and therefore developing the basis for efficient task realization within networked analytical infrastructures.

## 3. Method

The methodological approach of this research contribution follows the design-science-oriented method of Peffers et al. (2007).

Fig. 2 illustrates the procedure model as we suggest for improving inefficient task realization within organizations. The procedure model has been used for demonstration purposes in this contribution. One can observe 10 sequential phases and three possibilities for an iterative proceeding that enables a close to real-world simulation. Each phase of the procedure model is illustrated in the following.

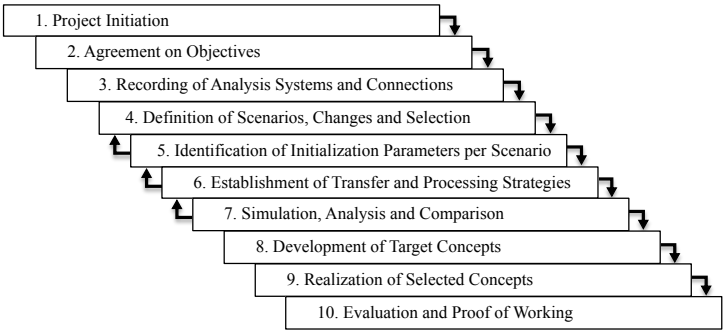



Figure 2: Procedure model

The *first phase* refers to the acquisition of projects, and includes contractual issues, sales, and marketing, among others. The *second phase* defines the focus of consecutive analyses. Based on a company strategic objectives and constraints, efficient task processing is identified. Within the *third phase*, the entire setting of analytical systems is recorded and mapped to given layers, which thus far are the CPS, local cloud, and public cloud levels. In the *fourth phase*, a collection of attractive scenarios is identified. Dependencies, similarities, and differences among scenarios are considered, so that they can be grouped into sequential transformations. The *fifth phase* serves to identify initialization parameters. Thereafter, in the *sixth phase*, the current customer transfer and processing strategy is established. The required parameters are collected in interviews with responsible analysis experts of the customer. The *seventh phase* realizes the simulation, which includes the transfer of scenarios, and customer-based and alternative transfer and processing strategies to the computational model. Based on a system-specific analysis, an analysis across systems, and a systematic comparison of focused approaches, the best candidates are identified and concluded regarding the current strategy. Within the *eighth phase*, insights from the seventh phase are used to develop concepts that improve the current customer situation. The *ninth phase* focuses on the implementation of selected target concepts. Hardware configuration adjustments, connections among systems, software modifications, and an adjustment in transfer and pro-

cessing strategies are included in this phase. The *tenth phase* evaluates whether the agreed objectives of the second phase could have been met, or further adjustments are required.

As the eighth phase results in insights that have to be verified, an *iterative* modification of previously set decisions is realized owing to *feedback circles* among the fourth, fifth, sixth, and seventh phases. Therefore, further scenarios, such as changes in initial connections, can be tested (fourth phase), initialization parameters can be changed, such as the use of stronger machines (fifth phase), or modified transfer strategies can be tested (sixth phase). Furthermore, feedback circles are required for validation purposes. Moreover, they are essential for highly evolutionary IT environments, as simulations consider changes.

## 4. Flexible Analytical Architecture Framework

In this section, we provide a mathematical model that will serve as a framework for the simulation and performance evaluation. The second sub-section presents the design of an NDM, which will be used for any system locally to decide whether an analytical task is processed on that system or a certain task is routed to another system. Based on this, a computational model is implemented, in which various task realization strategies can be simulated and their performance compared (subsection 3).

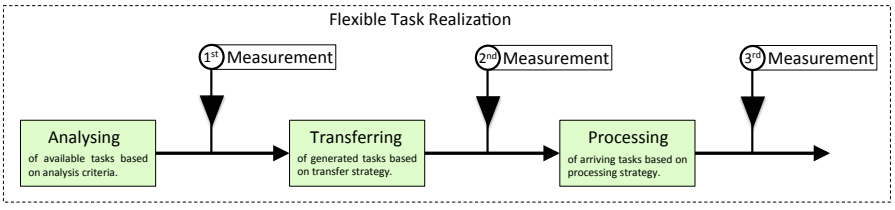



Figure 3: Process of flexible task realization

The interplay among the artifacts is organized in three steps (see Fig. 3), while process steps are visualized by green rectangles and control flow by arrows. Firstly, given available tasks, the situation must be analyzed with the

aid of analysis criteria. Then, task transfer can be carried out among the given analytical systems. Finally, the arriving tasks must be processed. Following any process step, a measurement can be carried out and provides for analyses.

### *4.1. Mathematical Model*

The following steps are realized in the mathematical model development. Firstly, the system and its parameters are characterized. Thereafter, inter-relationships are considered within the mathematical model. Furthermore, the optimization task of the current research is defined.

#### *4.1.1. System Characteristics and Parameters*

Firstly, parameters for the model development need to be defined. Consider a system with $N$ CPSs, labeled $C_i^{CPS},\ i = 1, 2, \cdots, N$. At the upper layer of the system, there are $M$ local clouds, labeled $C_j^{LC},\ j = 1, 2, \cdots, M$. Finally, at a layer above the local clouds there are $K$ public clouds, namely $C_1^{PC},\ C_2^{PC},\ \cdots,\ C_K^{PC}$. Fig. 1 (a) visualizes this 3D schematic of the system. The following are certain key assumptions that we need to work with:

1. We assume that all jobs at all levels are generated according to the Poisson distribution, and services follow the exponential distribution.
2. All the CPSs have buffers for storing jobs waiting to be processed, and the buffer sizes are unlimited.
3. Both the local and public clouds have unlimited buffers for holding jobs.

#### *4.1.2. Inter-relationships for Model*

At CPS $C_i^{CPS}$, let there be up to $N_i$ job types that can be generated, and each one arrives at a rate of $\lambda_{ij}^{CPS},\ j = 1, 2, \cdots, N_i$. We assume that jobs generated at CPS levels are independent; however, we can easily include dependencies at a later stage, if necessary. Jobs that are generated at a CPS level can be processed at that level or escalated to the local or public cloud level if required. The respective conditions are discussed later. Let the processing rate of a type $j$ job on $C_i^{CPS}$ be $\mu_{ij} = \mu_i^{CPS},\ \forall j$. By this, we are assuming that

the processing rates of all jobs are the same on a CPS; however, we distinguish the jobs only by their priorities.

At the local cloud level, jobs are generated at the rate of $\lambda_j^{LC},\ j = 1, 2, \cdots, M$ and we assume that only one type of job is generated at each local cloud. These jobs are independent of those generated at the CPS level. The processing rate of jobs at $C_i^{LC}$ is $\mu_i^{LC}$. However, there are also jobs that have been escalated from the CPS to LC level, and these will be discussed later.

Finally, at the public cloud level, jobs are generated at a rate of $\lambda_j^{PC}$, and we assume that only one job type is generated at the public cloud. These jobs are independent of those generated at the LC and CPS levels. The processing rate of jobs at $C_k^{PC}$ is $\mu_k^{PC},\ k = 1, 2, \cdots, K$, and there are also jobs that have been escalated from the CPS and LC levels to the PC level. These will also be discussed later. However, note that we do not have control over how, or even if, jobs move between different public clouds. Thus, we do not allow jobs to move from one public cloud to another. This is outside the scope of our work.

Jobs generated at the CPS levels may be processed at the CPS levels. However, some of these may be transferred to other CPSs for processing or escalated to an LC or even the PC. We have the following:

1. $p_{i,j,k}^{CPS \to CPS}$ is the ratio of $\lambda_{ij}^{CPS}$ moved to $C_k^{CPS},\ k \neq i$.
2. $p_{i,j,k}^{CPS \to LC}$ is the ratio of $\lambda_{ij}^{CPS}$ moved to $C_k^{LC}$.
3. $p_{i,j,k}^{CPS \to PC}$ is the ratio of $\lambda_{ij}^{CPS}$ moved to $C_k^{PC}$.

Note that we require that

$$\sum_{j=1}^{N_i}[\sum_{k=1;k\neq i}^{N} p_{i,j,k}^{CPS \to CPS} + \sum_{k=1}^{M} p_{i,j,k}^{CPS \to LC} + \sum_{k=1}^{K} p_{i,j,k}^{CPS \to PC}] \leq 1, i = 1, 2, ..., N. \quad (1)$$

If we define $\lambda_i^{CPS*}$ as the total amount of jobs arriving to be processed at $CPS_i$, then we have

$$\lambda_i^{CPS*} = \sum_{j=1}^{N_i}[1 - \sum_{k=1,k\neq i}^{N} p_{i,j,k}^{CPS \to CPS} - \sum_{k=1}^{M} p_{i,j,k}^{CPS \to LC} - p_{i,j}^{CPS \to PC}]\lambda_{ij}^{CPS}$$

$$+ \sum_{v=1,v\neq i}^{N} \sum_{j=1}^{N_v} p_{v,j,i}^{CPS\rightarrow CPS} \lambda_{v,j}^{CPS} + \sum_{v=1}^{M} \sum_{j=1}^{N_i} p_{v,j,i}^{LC\rightarrow CPS} \lambda_{v,j}^{LC} + \sum_{v=1}^{K} \sum_{j=1}^{N_i} p_{v,j,i}^{PC\rightarrow CPS} \lambda_{v,j}^{PC}. \tag{2}$$

Consider the local cloud level, and let

1. $p_{i,j}^{LC\rightarrow LC}$ be the ratio of jobs moved from $C_i^{LC}$ to $C_j^{LC}$, $\ i \neq j$;
2. $p_{i,j}^{LC\rightarrow PC}$ be the ratio of jobs escalated from $C_i^{LC}$ to $C_j^{PC}$; and
3. $p_{i,j,k}^{LC\rightarrow CPS}$ be the ratio of jobs de-escalated from $C_i^{LC}$ to $C_j^{CPS}$ and processed as a type $k$ job.

Furthermore, note that we require that

$$\sum_{j=1;j\neq i}^{M} p_{i,j}^{LC\rightarrow LC} + \sum_{j=1}^{K} p_{i,j,k}^{LC\rightarrow CPS} + \sum_{j=1}^{N} \sum_{k=1}^{N_i} p_{i,j}^{LC\rightarrow PC} \leq 1, i = 1, 2, ..., M. \tag{3}$$

Now, considering each $LC$, we assume that only one job type is processed there. Although the jobs may differ in types, we assume that escalated jobs have been scaled prior to escalation to normalize them, because the escalation cost has been considered. Let $\lambda_i^{LC*}$ be the arrival rate of jobs for processing at the local cloud $C_i^{LC}$.

$$\lambda_i^{LC*} = [1 - \sum_{j=1,j\neq i}^{M} p_{ij}^{LC\rightarrow LC} - \sum_{j=1}^{K} p_{i,j}^{LC\rightarrow PC} - \sum_{j=1}^{N} \sum_{k=1}^{N_i} p_{i,j,k}^{LC\rightarrow CPS}] \lambda_i^{LC}$$

$$+ \sum_{k=1,k\neq i}^{N} p_{k,i}^{CPS\rightarrow LC} (\sum_{v=1}^{N_k} \lambda_{k,v}^{CPS}) + \sum_{j=1,j\neq i}^{M} p_{j,i}^{LC\rightarrow LC} \lambda_j^{LC} + \sum_{j=1}^{M} p_{j,i}^{PC\rightarrow LC} \lambda_j^{PC}. \tag{4}$$

Finally, note that we require that

$$\sum_{j=1}^{M} P_{i,j}^{PC\rightarrow LC} + \sum_{j=1}^{N} \sum_{k=1}^{N_j} P_{i,j,k}^{PC\rightarrow CPS} \leq 1, \ i = 1, 2, \cdots, K. \tag{5}$$

Then, consider the public cloud level and let

1. $p_{i,j,k}^{PC\rightarrow CPS}$ be the ratio of jobs moved from $C_i^{PC}$ to $C_j^{CPS}$, $\ i \neq j$ as type $k$ jobs; and

2. $p_{i,j}^{PC \to LC}$ be the ratio of jobs de-escalated from $C_i^{PC}$ to $C_j^{LC}$.

Note that we cannot move jobs between public clouds, as that is managed by a third part. For the public cloud, the rate at which jobs arrive for processing is

$$\lambda_i^{PC*} = \lambda_i^{PC} + \sum_{k=1}^{N} \sum_{j=1}^{N_k} p_{k,j,i}^{CPS \to PC} \lambda_{k,j}^{CPS} + \sum_{k=1}^{M} p_{k,i}^{LC \to PC} \lambda_k^{LC}$$

$$- [\sum_{i=1}^{M} p_{k,i}^{PC \to LC} + \sum_{j=1}^{N} \sum_{k=1}^{N_j} p_{i,j,k}^{PC \to CPS}] \lambda_i^{LC}. \quad (6)$$

The queueing model representing this system is actually a multiple set of single-node heterogeneous queues in parallel, with possible job transfers between queues. In order for the system to be stable, we require that each be stable; hence, we require that

$$\max \left\{ \frac{\lambda_i^{CPS*}}{\mu_i^{CPS}}, i = 1, 2, \cdots, N; \ \frac{\lambda_\ell^{LC*}}{\mu_\ell^{LC}}, \ \ell = 1, 2, \cdots, M; \ \frac{\lambda_k^{PC*}}{\mu_k^{PC}}, \ k = 1, 2, \cdots, K \right\} < 1. \quad (7)$$

When dealing with the compact job type rates, several jobs of a certain type are compressed to one rate, and the arrival rates $\lambda^J$ can serve for the derivation of performance criteria, as follows.

Let $\omega_{i,j,k} = \frac{\lambda_{i,j}^J p_{i,j,k}^J}{\lambda_{i,j}^{J*}}$ be the percentage of arriving jobs at a system $k$ from system $i$, and job types $j$, which are all on the CPS level $M$, on local cloud level $N$, and on public cloud level $K$ systems. If we define

1. $g_{k,j}^*$ as the mean generation time after the transfer of jobs on system $k$ and job type $j$;
2. $r_{k,j}^*$ as the mean remaining time after the transfer of jobs on system $k$ and job type $j$;
3. $im_{k,j}^*$ as the mean importance after the transfer of jobs on system $k$ and job type $j$; and
4. $d_{k,j}^*$ as the mean divisibility after the transfer of jobs on system $k$ and job type $j$;

then, following transfer, we have

$$g^*_{k,j} = \sum_{i=1}^{M+N+K} \omega_{i,j,k} \cdot g_{k,j}, \tag{8}$$

$$r^*_{k,j} = \sum_{i=1}^{M+N+K} \omega_{i,j,k} \cdot r_{k,j}, \tag{9}$$

$$im^*_{k,j} = \sum_{i=1}^{M+N+K} \omega_{i,j,k} \cdot im_{k,j}, \tag{10}$$

$$d^*_{k,j} = \sum_{i=1}^{M+N+K} \omega_{i,j,k} \cdot d_{k,j}. \tag{11}$$

*4.1.3. Optimal Strategy for Escalation*

The rates at which jobs are generated are not within our control. Hence, all of the arrival rates, $\lambda^{CPS}_{i,j,k}$, $\lambda^{LC}_{j}$, and $\lambda^{PC}_{k}$, are fixed or preset. Similarly, the processing rates at the nodes are predetermined. However, we can influence which portions of the jobs are escalated or sent to equivalent nodes. Thus, for control purposes, we have the ratios as decision variables. Let us define the following vectors:

$$\mathbf{p}^{CPS\to CPS} = \{p^{CPS\to CPS}_{i,j,k},\ \forall(i,j,k)\},$$

$$\mathbf{p}^{CPS\to LC} = \{p^{CPS\to LC}_{i,j,k},\ \forall(i,j,k)\},$$

$$\mathbf{p}^{CPS\to PC} = \{p^{CPS\to PC}_{i,j},\ \forall(i,j)\},$$

$$\mathbf{p}^{LC\to CPS} = \{p^{LC\to CPS}_{i,j,k},\ \forall(i,j,k)\},$$

$$\mathbf{p}^{LC\to LC} = \{p^{LC\to LC}_{i,j},\ \forall(i,j)\},$$

$$\mathbf{p}^{LC\to PC} = \{p^{LC\to PC}_{i,j},\ \forall(i,j)\},$$

$$\mathbf{p}^{PC\to CPS} = \{p^{PC\to CPS}_{i,j},\ \forall(i,j,k)\},$$

$$\mathbf{p}^{PC\to LC} = \{p^{PC\to LC}_{i,j},\ \forall(i,j)\}.$$

Now, let us define the following vectors:

$$\mathbf{P}^{CPS} = [\mathbf{p}^{CPS\rightarrow CPS},\ \mathbf{p}^{CPS\rightarrow LC},\ \mathbf{p}^{CPS\rightarrow PC}],$$

$$\mathbf{P}^{LC} = [\mathbf{p}^{LC\rightarrow CPS},\ \mathbf{p}^{LC\rightarrow LC},\ \mathbf{p}^{LC\rightarrow PC}],$$

$$\mathbf{P}^{PC} = [\mathbf{p}^{PC\rightarrow CPS},\ \mathbf{p}^{PC\rightarrow LC}],$$

$$\mathbf{p} = [\mathbf{p}^{CPS}\ \mathbf{p}^{LC},\ \mathbf{p}^{PC}]. \tag{12}$$

Once we decide on an objective function, such as minimizing the total cost or any other measure, which we term $\mathbf{f}(\mathbf{p})$, we can perform an optimization problem, as follows:

$$min_{\mathbf{p}} f(\mathbf{p}), \tag{13}$$

$$s.t.\quad (1), (2)\ \ and\ \ (3), \tag{14}$$

$$0 \leq \mathbf{p} \leq 1. \tag{15}$$

Eq. (13) implies that we are attempting to select the vector $\mathbf{p}$ that minimizes the function $\mathbf{f}(\mathbf{p})$. Eq. (14) implies that we wish to achieve this subject to meeting the conditions of the ratios in Eq. (1), (2), and (3), as well as the stability condition of Eq. (7). Finally, Eq. (15) simply implies that the ratios must satisfy the non-negativity conditions and must be less than 1.

### *4.2. New Decision Maxim*

In this section, an NDM is provided for identifying the most suitable selection within the computing framework. In the first step, basic options are presented. Thereafter, allocation criteria are presented and their interplay with an escalation prioritization in the form of weights is defined. These could be assigned in order to meet the customer-specific requirements and optimization selections.

Table 2: Available allocation options

| **Option / Level** | **Vertical-up** | **Vertical-down** | **Horizontal** | **Local** |
|---|---|---|---|---|
| CPS level | X | - | X | X |
| Local cloud | X | X | X | X |
| Public cloud | - | X | (X) | X |

*4.2.1. Basic Distribution Options*

When faced with an analytical task, a system has four basic options regarding the task distribution within the analytical infrastructures. These can be grouped into three different categories, as follows.

Firstly, tasks can be computed on a *local* level. Here, no transfer will be realized at all, and the jobs remain at the computing or scheduled unit.

Secondly, there are *vertical escalations*: here, an *upward escalation* refers to the transfer of jobs from computing units to a higher and probably more powerful level. Of course, this is not an option for systems at the very top level. Furthermore, a *downwards reallocation* refers to transfers from computing units to lower level systems, which cannot be applied at the very bottom levels.

Thirdly, tasks can be computed on neighboring systems, which is known as *horizontal distribution* from here on. According to the different levels and distribution options, Table 2 visualizes the limitations of high- and low-level systems.

*4.2.2. Allocation Process*

Within the NDM, $c_l$ criteria can be used to identify the most preferable allocation option $m$, with $l = 1, ..., L$ and $m = 1, ..., M$. In this work, only four evaluation criteria are exemplary illustrated in order to demonstrate the approach; hence, $L = 4$. These are task effort, priority, divisibility, and current load as follows: The *task effort* $c_1 = \rho_i^J$ (see Eq. 19) focuses on the task and the processing unit that is going to deal with that task, and considers its relative performance. Hence, the effort represents the local task computing capabilities. The *priority* is the relevance of a task to the global company goal of the company (e.g., production process). This includes the importance of a task result $c_2 =$

$im_{k,j}$ (see Eq. 10) and the remaining time $r_{k,j}$ (see Eq. 9) until the result is required. The *Divisibility* $c_3 = d_{k,j}$ (see Eq. 11) refers to the aspect of how far a task can be split among different computing units. A task with low divisibility cannot be effectively computed in parallel on multiple computing units. A task with a high divisibility can easily be distributed among multiple computing units. The *current load* $c_4 = w_i^J$ (see Eq. 23-25) is a dynamic parameter referring to the current free capacities of systems to compute additional analytical tasks.

In terms of the prioritization example of Fig. 4, an example task is characterized with a task effort of 0.9, divisibility of 0.2, priority of 0.5, and current load of 0.5. The available allocation options $m$ of Table 2 can be weighted with regard to the selected evaluation criteria $c_l$ by a customer. Hence, the customer escalation prioritization of each allocation option can be found in Fig. 4 on the right, which shows 16 prioritization weights $p_{l,m}$. The option-specific evaluation can then be realized following

$$Eval_m = \sum_{l=1}^{L} c_l \cdot p_{l,m}. \tag{16}$$

Then, the most suitable allocation option can be identified by means of

$$max_m(Eval_m). \tag{17}$$

As can be observed in Fig. 4, the most appropriate option for the current example task is a vertical upward escalation.

### *4.3. Computational Model*

The computational model transfers the mathematical model of section 4.1 to a computational simulation model. The described optimization tasks consist of two decisions, which both affect the performance measurement: decisions that refer to the transfer of tasks within the systems, and decisions that refer to the order in which analytical tasks are processed following transfer in individual systems. The following demonstrates the manner in which strategies of efficient

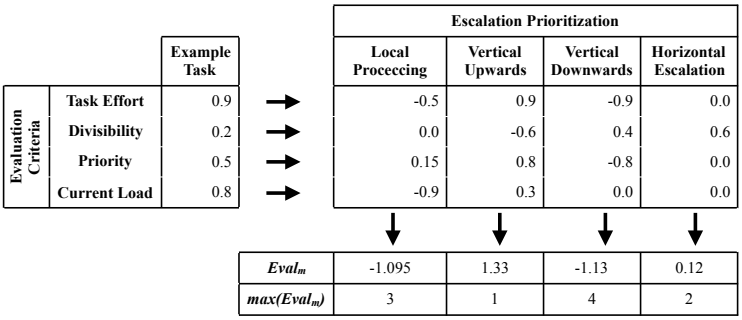



Figure 4: Prioritization Example.

task realization approaches, as presented in section 2.2, are transferred to the computational model. Transfers within the network of systems are realized according to the following approaches.

- *No-transfers-at-all*: As no transfers are realized; for example, a company does not care about transfers, this strategy is realistic. It servers as a reference point that will be optimized by intelligent task transfers.
- *Workshop-based-transfers*: This strategy is generated by an analytical team within a workshop, as required by the sixth phase of the procedure model (section 3).
- *New-decision-maxim*: The strategy suggested in section 4.2.

The processing of arrived task types in each system within the network of systems is realized by the approaches displayed in Table 3.

In the above, 12 processing strategies are displayed, as they were plausible according to common strategies and some were applied by customers as the sixth phase of the procedure model (section 3).

## 5. Evaluation and Discussion

Building on the mathematical model of section 4.1 and its computational implementation in section 4.3 using Python 2.7, the following demonstrates the

Table 3: Overview of processing strategies.

| **Id) Processing strategy** | **Focus of ordering** |
|---|---|
| 1) *Alphabetically ascending* | Tasks with smallest *id* are realized first. |
| 2) *Alphabetically descending* | Reversed order of previous approach. |
| 3) *First-in-first-out* | Tasks with the smallest $\lambda^{J*}$ are realized first. |
| 4) *Last-in-first-out* | Reversed order of previous approach. |
| 5) *First-remaining-in-first-out* | Tasks with smallest $r^{J*}$ are realized first. |
| 6) *Last-remaining-in-first-out* | Reversed order of previous approach. |
| 7) *High-importance-in-first-out* | Tasks with smallest $im^{J*}$ are realized first. |
| 8) *Low-importance-in-first-out* | Reversed order of previous approach. |
| 9) *Fastest-in-first-out* | Tasks with smallest $\mu^{J*}$ are realized first. |
| 10) *Slowest-in-first-out* | Reversed order of previous approach. |
| 11) *Cheapest-in-first-out* | Tasks with smallest $c^{J*}$ are realized first. |
| 12) *Most-expensive-in-first-out* | Reversed order of previous approach. |

simulation results and describes the NDM application as discussed in section 4.2. As the NDM is displayed alongside further strategies, a benchmark can be created in order to compare the approaches. Hence, the first subsection designs a performance evaluation framework that is used for the benchmark. As the second subsection demonstrates artifacts following the procedure model of section 3, this clarifies its application, and the results are structured by its phases. The third subsection discusses the simulation results regarding a performance analysis. Therefore, the simulation results presented here serve as a proof of concept.

### *5.1. Performance Evaluation Framework*

A framework for the measurement of the observed decision strategy performances is developed, as follows. Firstly, key performance indicators are defined. Thereafter, a common objective function is established. These are applied in a demonstration and are the foundation of its evaluation.

#### *5.1.1. System-specific Performance Measures*

Given the description of the system model, its characteristics and assumptions make it clear that the system is actually a multiple set of single-node heterogeneous queues connected in parallel, with possible job transfers between queues.

Let $c_{i,j}^{J*}$ be the costs for the realization of task type $j$ of system $i$ at level $J$, which may be the CPS, local cloud or public cloud level:

$$c_{i,j}^{J*} = \lambda_{i,j}^{J*} c_{i,j}^{J}. \tag{18}$$

Let $\rho_i^J$ be the traffic intensity for system $i$ on level $J$, which may be the CPS, local cloud or public cloud level:

$$\rho_i^J = \frac{\lambda_i^{J*}}{\mu_i^J}. \tag{19}$$

If we define:r

1. $l_i^{CPS}$ and $w_i^{CPS}$ as the mean number of jobs waiting, and the mean waiting times of jobs, respectively, at $C_i^{CPS},\ i = 1, 2, \cdots, N$. These are independent of which job types class they are; i.e., we have lumped them all together;
2. $l_i^{LC}$ and $w_i^{LC}$ as the mean number of jobs waiting, and the mean waiting times of jobs, respectively, at $C_i^{LC},\ i = 1, 2, \cdots, M$. These are independent of which job types class they are; i.e., we have lumped them all together;
3. $l_i^{PC}$ and $w_i^{PC}$ as the mean number of jobs waiting, and the mean waiting times of jobs, respectively, at $C_i^{PC},\ i = 1, 2, \cdots, K$. These are independent of which job types class they are; i.e., we have lumped them all together.

Then, using the queuing results from Gross et al. (2008), we have

$$l_i^{CPS} = \frac{\rho_i^{CPS}}{1 - \rho_i^{CPS}},\ i = 1, 2, \cdots, N, \tag{20}$$

$$l_j^{LC} = \frac{\rho_j^{LC}}{1 - \rho_j^{LC}},\ j = 1, 2, \cdots, M, \tag{21}$$

$$l_k^{PC} = \frac{\rho_k^{PC}}{1 - \rho_k^{PC}},\ k = 1, 2, \cdots, K. \tag{22}$$

$$w_i^{CPS} = (\mu_i - \lambda_i^{CPS*})^{-1},\ i = 1, 2, \cdots, N, \tag{23}$$

$$w_j^{LC} = (\mu_j - \lambda_j^{LC*})^{-1},\ j = 1, 2, \cdots, M, \tag{24}$$

$$w_k^{PC} = (\mu_k - \lambda_k^{PC*})^{-1},\ k = 1, 2, \cdots, K. \tag{25}$$

Using Eqs. (1) to (14), we can determine the performance of our system, given the system parameters at a given time step $t$.

Whichever decision we take at any time step should be guided, in the long run, by the results of our steady-state (stable) model. For example, if our optimal $P_{ij}^{LC \to PC} = 0.1$, at any time period $(t_1, t_2)$, only 10% of jobs from local cloud $i$ will be sent to public cloud $j$, regardless of how the 10% is achieved.

*5.1.2. Derivation of Objective Function*

Focusing on the optimization problem of Eq. (13), an optimization may consider several influencing factors. Based on the design-oriented approach of Grum et al. (2017), the following refers to a set of factors validated in three ways by experts providing a high level of expertise in analytical infrastructures. This set can be found in the following.

- The waiting time $w_{i,j}^t$ refers to the period that begins at the time step at which a job part $j$ occurred at a CPS $i$ and ends at the current time step $t$.

- The remaining time $r_j^t$ refers to the period that begins at the current time step $t$ and ends at that time step, where the result of a current job part $j$ is required.

- The processing cost $c_{i,j}^t$ refers to the costs that the processing of job part $j$ incurs at CPS $i$. As one assumes these to develop over time because of market mechanisms, they are time dependent, and we initially assume these to be constant.

- The importance $im_j^t$ of a job part $j$ refers to the criticality of a current job part at time step $t$. As one assumes the system to be flexible because of changing customer requirements, these are time dependent, and we initially assume these to be constant.

- The transfer cost $tr^t_{i,j,k}$ refers to the costs incurred because of the transfer of job part $j$ from CPS $i$ to CPS $k$ at time step $t$. As one assumes these to develop over time because of current bottlenecks, they are time dependent, and we initially assume these to be constant.

The assessment of efficient task processing is based on the empirically validated objectives of Grum et al. Grum et al. (2017), resulting in the use of Eq. (26).

$$f(\mathbf{p}) = \sum_{t=1}^{o} \sum_{i=1}^{n} \sum_{j=1}^{m_i} a^t_{i,j} p^t_{i,j}, \tag{26}$$

where $o$ is time steps, $n = N + M + K$, $m_i$ is the number of parts of job $i$, and

$$a^t_{i,j} = im^t_j \ \frac{w^t_{i,j}}{r^t_j} \ \frac{c^t_{i,j}}{min(tr^t_{i,j,k})}. \tag{27}$$

As only the processing rates $p^t_{i,j}$ of each system $i$ can be controlled for job part $j$ at each time step $t$, the objective function to be optimized would be Eq. (13).

*5.1.3. Performance Measures Across Systems*

In addition to the global performance measurement criteria of Eq. (26), the following demonstrates how the given criteria can be used for performance evaluation across all systems. These may provide the foundation for benchmarks.

Considering the costs as they have been used in Eq. (27), costs at a system-specific level are accumulated in order to obtain the costs at a level across systems:

$$C^{J*}_{i,j} = \sum_{i=1}^{M+N+K} \sum_{j=1} c^{J*}_{i,j}. \tag{28}$$

Considering the mean number of waiting jobs of Eqs. (20) to (22) at a system-specific level, the mean number of waiting jobs at a level across systems is

$$L^{J*}_{i,j} = \sum_{i=1}^{M+N+K} \sum_{j=1} l^{J*}_{i,j}. \tag{29}$$

Considering the mean waiting time of jobs of Eqs. (23) to (25) at a system-

specific level, the mean number of waiting jobs at a level across systems is

$$W_{i,j}^{J*} = \sum_{i=1}^{M+N+K} \sum_{j=1} w_{i,j}^{J*}. \tag{30}$$

Considering the traffic intensity of jobs of Eq. (19) at a system-specific level, the traffic intensity at a level across systems is

$$\rho_{i,j}^{J*} = \sum_{i=1}^{M+N+K} \sum_{j=1} \rho_{i,j}^{J*}/(M+N+K). \tag{31}$$

*5.2. Demonstration*

As demanded by the methodological approach of section 3, a customer was included in a workshop session. Here, three process experts, five analytical experts, and two production experts were integrated and faced with two examples: firstly, a small numerical example considering only four systems in order to realize a quick understanding and trusted level of working with the NDM, as presented in the following; and secondly, their company-specific production setting, which is not presented here.

*1) Project initiation:* The first example is a theoretical example that was created with the intention of developing a small, easily understandable simulation setting.

*2) Objectives:* The general objective was to compare the customer current task realization approach (workshop-based strategies) with NDM-based strategies. Furthermore, task realization with no intervention was achieved as a reference point (no-transfers-at-all).

*3) Systems and connections:* For illustrative purposes, consider a system with two $CPSs$: one robot and one printer; that is, $M = 2$. Assume that each CPS has only one job type, and that we have only one local cloud $N = 1$ and one public cloud $K = 1$. The one-sided connections among all systems are drawn as Fig. 5 intends to visualize with the aid of arrows. The direction of allowed transfers is indicated by the visualized arrows.

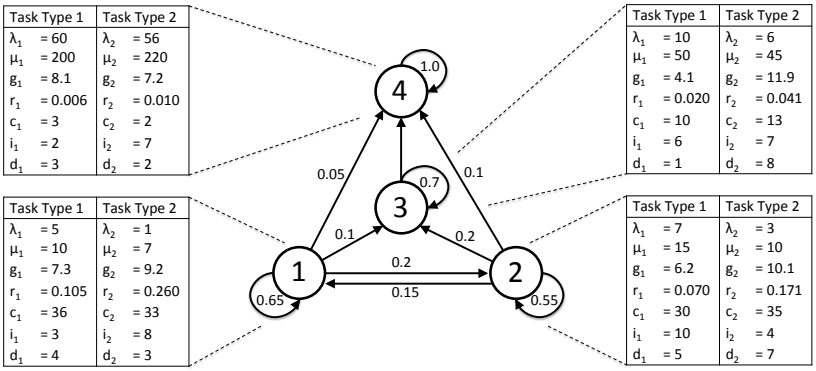



Figure 5: Initialization of numerical example

*4) Scenario collection:* The following two scenarios are selected for consecutive phases.

a) The first scenario includes a transfer strategy that demonstrates imbalances.

b) The second scenario compares all three transfer strategies in combination with all 12 processing strategies as defined in section 4.3.

As these scenarios indicate neither a temporal nor any further relation, they can be realized independently (both show only a single time step, $t = 0$).

*5a) Initialization parameter:* At present, we only consider one task type. Let $\lambda_1^{CPS} = 5,\ \lambda_2^{CPS} = 7,\ \mu_1^{CPS} = 10$, and $\mu_2^{CPS} = 15$. Throughout this example, all arrivals and services are rates per unit time. Let us assume only one local cloud with $\lambda^{LC} = 10,\ \mu^{LC} = 50$ and one public cloud with $\lambda^{PC} = 60,\ \mu^{PC} = 200$. Let us further assume that $\mu_1^{CPS} = 10,\ \mu_2^{CPS} = 15,\ \mu^{LC} = 50$ and $\mu^{PC} = 200$. Further initialization, parameters such as $g^J,\ r^J,\ c^J,\ im^J$, and $^J$ can be found in Fig. 5 alongside the corresponding system. We do not include subscripts when there is only one unit in order to avoid unnecessarily complicated notations.

*6a) Transfer and processing strategies:* Suppose we have a predetermined transfer strategy with $p_{1,2}^{CPS\to CPS} = 0.2, p_{1,1}^{CPS\to LCP} = 0.1, p_{1,1}^{CPS\to PC} = 0.05$ for the first CPS; for the second CPS, we have $p_{2,1}^{CPS\to CPS} = 0.15, p_{2,1}^{CPS\to LCP} = 0.2, p_{2,1}^{CPS\to PC} = 0.1$; and finally, for the local cloud, we have $p_{1,1}^{LC\to PC} = 0.3$.

*7a) Simulation, analysis, and comparison:* Applying the results from the equations above, we have

$$\lambda_1^{CPS*} = 4.30,\ \lambda_2^{CPS*} = 4.85,\ \lambda^{LC*} = 8.9,\ \lambda^{PC*} = 63.95.$$

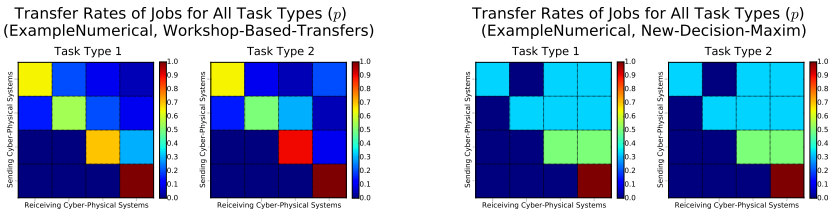
Since $\mu_1^{CPS} = 10,\ \mu_2^{CPS} = 15,\ \mu^{LC} = 50$ and $\mu^{PC} = 200$, and we observe that the stability conditions are met with $\rho_1^{CPS} = 0.43,\ \rho_2^{CPS} = 0.3233,\ \rho^{LC} = 0.178$, and $\rho^{PC} = 0.31975$. Hence, we obtain $l_1^{CPS*} = 0.75,\ l_2^{CPS*} = 0.48,\ l^{LC*} = 0.22$, and $l^{PC*} = 0.47$. Moreover, $w_1^{CPS*} = 0.175,\ w_2^{CPS*} = 0.099,\ w^{LC*} = 0.024$, and $w^{PC*} = 0.007$, all of which are in units of time. If the arrivals are in numbers per second, the waiting times are in seconds per item.

Suppose the processing rate at $C_1^{CPS}$ is actually $\mu_1^{CPS} = 5.0$; then, we observe that $\rho_1^{CPS} = 0.86$, and as a result, we will obtain $l_1^{CPS*} = 6.143$ and $w_1^{CPS*} = 1.429$. This immediately indicates an imbalance in the system, as the performance at $C_1^{CPS}$ is not as effective. In this case, we may need to change our escalation rates or ratios. With the aid of a small example, we have demonstrated that this model can be used to obtain the performance of a preset escalation strategy. The second scenario focuses on the manner in which to plan the escalation optimally.

*5b) Initialization parameter:* Now, we consider two task types. Let us assume the initialization parameters to be as found in Fig. 5 alongside the corresponding system.

*6b) Transfer and processing strategies:* Suppose we have three transfer strategies, as follows. The first transfer matrix of the "no-transfers-at-all" strategy resembles an identity matrix, as 100% of each task type remains at its origin system at position $i = j$. The second transfer matrix of the "workshop-based" strategy has been established by the experts previously mentioned, as required by the procedure model in phase six (section 3). A visualization can be found in Fig. 6(a). Single transitions can be found in Fig. 5, alongside the corresponding arrows. It should be noted that the same transfer parameters have been used for both task types. The third transfer matrix, originating from the "new-decision-maxim," can be found in Fig. 6(b).

*7b) Simulation, analysis, and comparison:* The initial configuration results




(a) Workshop-Based. (b) New-Decision-Maxim-Based.

Figure 6: Transfer strategy of numerical example

in a workload $\rho_{i,j}^{J*}$ (see Eq. 31) can be seen in Fig. 7. Here, it can be observed that the traffic intensity of any system does not exceed the limit of 1.0. Hence, no system is overloaded, otherwise it would break down. As the illustrated traffic intensity is not changed by transfers of the first transfer strategy (no-transfers-at-all) in this case, its workload following transfers is the same, and a transfer is not required. Focusing on the traffic intensity of all *LC1* and *PC1*, one can identify the greatest potential in free capacities, although both CPSs provide free capacities as well.

As jobs are transferred, the initial workload is changed, and the optimal task realization runs for each transfer strategy can be observed in Fig. 8. The optimal runs are characterized by the objective function, as presented in section 5.1.2.

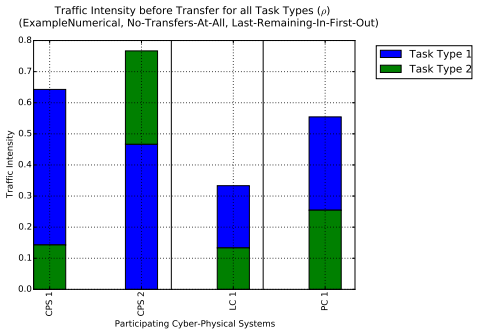
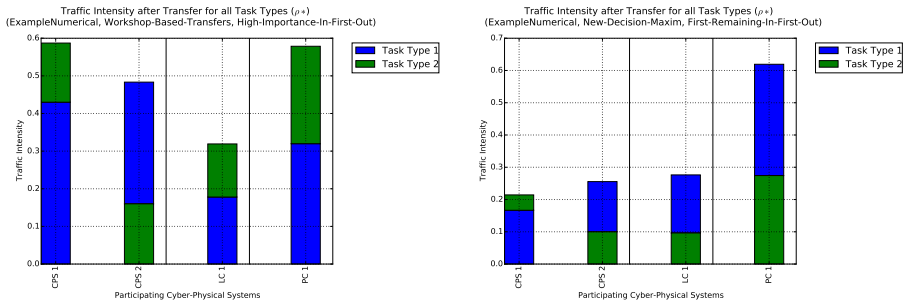



Figure 7: Initial traffic intensity of numerical example




(a) Workshop-based
- High-importance-in-first-out

(b) New-decision-maxim-based
- First-remaining-in-first-out

Figure 8: Traffic intensity of optimal task realization runs of numerical example

The visualization on the left illustrates the system-specific traffic intensities following workshop-based transfers. Beginning with the processing of tasks with the highest priority (high-priority-in-first-out), one can observe a significant transfer of tasks of *CPS2* to *PC1* with increased computing power, so that prioritized task realization can be achieved. As no system is overloaded and the constraints were considered fairly effectively, the experts established a working strategy. The visualization on the right illustrates the system-specific traffic intensities following NDM-based transfers. Beginning with the processing of tasks with the lowest remaining time (first-remaining-in-first-out), one can observe a significant transfer of tasks belonging to weak systems to systems with increased computing power. Here, rapid task realization of tasks with short reaction times can be achieved, and no system is overloaded in this case either.

*8) Target concepts:* Based on a comparing of the results of phase 7b), a change to the NDM is suggested. In this case, the processing strategy demonstrates minor importance, as all processing strategy combinations and the new decision strategy improve the customer situation. A detailed explanation can be found in the following section.

*9-10) Further phases:* Consecutive phases have not been considered because of the theoretical nature of the numerical example.

*5.3. Performance Analysis*

Building on the demonstration of section 5.2, the following evaluates the simulated task realization strategies and creates a trade-off among various approaches. The simulations result in different performance levels for each transfer strategy. An overview is provided in Table 4.

Table 4: Transfer strategy performance of numerical example

| **Transfer strategy** | **New-decision-maxim** | **Workshop-based-transfers** | **No-transfers-at-all** |
|---|---|---|---|
| Total processing costs | 4633.333 | 4379.4500 | 4222.0000 |
| Total traffic intensity | 0.3413 | 0.4921 | 0.5744 |
| Total number of waiting jobs | 1.7760 | 2.8093 | 3.6441 |
| Total time of waiting jobs | 0.5229 | 0.6262 | 0.6984 |
| Total job realization with time | {7, 8} | {5, 6} | {2, 3, 4} |

It can be observed that the most expensive, system-wide task realization is caused by NDM transfers, followed by workshop-based transfers and no-transfers-at-all. Faced with the total number of jobs realized with time, the additional costs can be justified. NDM-based transfers realized the highest number of tasks with time, followed by workshop-based transfers and no-transfers-at-all. Illustrating 4 systems in the example with 2 job types per system, all 8 jobs types could only have been realized with time following the NDM, while 5 or 6 out of 8 job types could have been realized following workshop-based transfers. Only $2, 3, 4$ out of 8 job types could have been realized with time following the no-transfers-at-all transfer strategy. In general, superior results were achieved because of the transfer focus on more powerful systems. This is why the further key performance indicators improve, and the number of waiting jobs and total time of waiting jobs decrease with superior approaches.

Different performance levels could be identified when considering all processing strategies. Table 5 displays the task realization strategies (including transfer and processing strategies) sorted by the objective function of Eq. 27. Since this is similar to the sorting on base of the number of task types realized in time (first criterion) and the total job remaining time (second criterion), this gives evidence for the objective function working.

NDM-based realization strategies can be identified as superior without exception, followed by workshop-based realization strategies without exception.

Table 5: Process strategy performance of numerical example.

| Transfer strategy | Process strategy | Com. obj. func. ↑ | Total job remaining time | Total job realization with time ↓ |
|---|---|---|---|---|
| N-D-M | First-remaining-in-first-out | 0.5378 | 0.4177 | 8 |
| N-D-M | High-importance-in-first-out | 0.5379 | 0.4163 | 8 |
| N-D-M | Cheapest-in-first-out | 0.5379 | 0.4163 | 8 |
| N-D-M | Task-type-descending | 0.5386 | 0.3877 | 8 |
| N-D-M | Last-in-first-out | 0.5387 | 0.3863 | 8 |
| N-D-M | Fastest-in-first-out | 0.5387 | 0.3863 | 8 |
| N-D-M | First-in-first-out | 0.6029 | 0.4691 | 7 |
| N-D-M | Slowest-in-first-out | 0.6029 | 0.4691 | 7 |
| N-D-M | Task-type-ascending | 0.6030 | 0.4678 | 7 |
| N-D-M | Low-importance-in-first-out | 0.6038 | 0.4391 | 7 |
| N-D-M | Most-expensive-in-first-out | 0.6038 | 0.4391 | 7 |
| N-D-M | Last-remaining-in-first-out | 0.6039 | 0.4378 | 7 |
| W-B-T | High-importance-in-first-out | 1.1769 | 0.1979 | 6 |
| W-B-T | Cheapest-in-first-out | 1.1779 | 0.1964 | 6 |
| W-B-T | Task-type-descending | 1.2000 | 0.2032 | 6 |
| W-B-T | Last-in-first-out | 1.2002 | 0.2001 | 6 |
| W-B-T | First-remaining-in-first-out | 1.2002 | 0.2001 | 6 |
| W-B-T | Fastest-in-first-out | 1.2002 | 0.2001 | 6 |
| W-B-T | First-in-first-out | 1.3712 | 0.2594 | 5 |
| W-B-T | Last-remaining-in-first-out | 1.3712 | 0.2594 | 5 |
| W-B-T | Slowest-in-first-out | 1.3712 | 0.2594 | 5 |
| W-B-T | Task-type-ascending | 1.3714 | 0.2563 | 5 |
| W-B-T | Most-expensive-in-first-out | 1.3972 | 0.2631 | 5 |
| W-B-T | Low-importance-in-first-out | 1.3984 | 0.2616 | 5 |
| N-T-A-A | Last-remaining-in-first-out | 1.8987 | 0.2466 | 4 |
| N-T-A-A | Low-importance-in-first-out | 1.8987 | 0.2466 | 4 |
| N-T-A-A | Task-type-descending | 1.9029 | 0.1897 | 4 |
| N-T-A-A | First-in-first-out | 2.3549 | 0.1579 | 3 |
| N-T-A-A | Slowest-in-first-out | 2.3549 | 0.1579 | 3 |
| N-T-A-A | Cheapest-in-first-out | 2.3633 | 0.1135 | 3 |
| N-T-A-A | Most-expensive-in-first-out | 2.3707 | 0.2312 | 3 |
| N-T-A-A | Last-in-first-out | 2.3791 | 0.1868 | 3 |
| N-T-A-A | Fastest-in-first-out | 2.3791 | 0.1868 | 3 |
| N-T-A-A | Task-type-ascending | 3.1405 | 0.1550 | 2 |
| N-T-A-A | First-remaining-in-first-out | 3.1474 | 0.0981 | 2 |
| N-T-A-A | High-importance-in-first-out | 3.1474 | 0.0981 | 2 |

Finally, realization strategies based of no-transfers-at-all occur. While a concrete ranking of all task realization strategies can be observed in the table, the superior strategies focus on the remaining time, importance, and processing costs. Within the provided visualizations, only the optimal candidates per category are considered. Detailed insights into every approach can be found in the attached files.

## 6. Conclusions

*Critical appraisal.* In accordance with the design-science research guidelines of Hevner et al. (2004), this contribution satisfies the requirements for effective design-science research and is complete, as indicated in Table 6.

Table 6: Design-science research guidelines

| Guideline | Description |
|---|---|
| Guideline 1: Design as an artifact | The authors design a flexible architectural framework described by a mathematical formulation and connected to a performance evaluation framework. Both frameworks are implemented as a computational model. An NDM is designed that is benchmarked in simulations with existing task realization approaches. |
| Guideline 2: Problem relevance | Considering the previously mentioned artifacts, the business problem of complex task realization approaches is overcome by a simple heuristic. A common framework is presented that can serve for comparison with further task realization approaches, and be applied to different analytical infrastructures. As the framework is parameter-based, the concrete framework around contemporary IT is reasonable, given a highly evolutionary environment and the continual application of the framework. |
| Guideline 3: Design evaluation | The efficacy of the designed artifacts was demonstrated rigorously by means of simulations. The utility and quality of the NDM was demonstrated by a performance evaluation and benchmarks. The execution precisely followed the documented mathematical expressions and quantitative criteria. Therefore, validation of the theoretical model and simulation are valid within a theoretical, small example, and will be transferred to a larger example in a real-life setting as a next step. |
| Guideline 4: Research contributions | The design-science contributions of this research are the proposed NDM and evaluation results in the form of a simulation and performance analysis. These contributions advance our understanding of the manner in which to optimize analytical tasks within modern dynamic analytical infrastructures. In regard to related works of Tab. 1, this integrates relates works, satisfies categories identified here and contributes to close the research gab. |
| Guideline 5: Research rigor | Research on task realization strategies has long been based on complex algorithms, such has Hadoop systems, edge computing or cloud computing (Lopez et al. (2015)). In this contribution, heuristics provide the underlying task realization strategy, which allows for efficient task realizations (such as processing costs, traffic intensity, total number of waiting jobs, total time of waiting jobs, and total job realization with time) and enables the development of more context-specific strategies, benchmarks, and applications. |
| Guideline 6: Design as a search process | As discussed previously, the implementation of task realization strategies, application, and benchmarking in analytical environments in iterations is essential. The authors studied variations in realization strategies over a period of 14 months within the aforementioned company workshops. Creativity and problem-solving capabilities were involved in the construction of an NDM. |
| Guideline 7: Communication of research | The presentation of this research is aimed at an audience familiar with queuing theory and analytics. Even so, the contribution provides useful information for managerial audiences. While the authors present a thorough discussion of economic performance criteria, the contribution provides evidence for both technical implementations and economic reasoning. |

A major contribution lies in the integration of various domain-specific knowledge bases (Tab. 1) in one coherent simulation model. This provides a basis for the comparison of efficient task realization strategies in analytical infrastructures, and provides an answer to the first research question. Through its application in a demonstration scenario, common strategies are evaluated with regard to their efficiency (second research question). As a NDM was developed, the strategy variants of which outperformed common approaches in major points, helpful guidelines regarding infrastructure optimization can be derived for both managers and technical experts.

*Limitations and lessons learned.* Although the results demonstrate clear dominance of the NDM, this cannot yet stand as a generalization since the applied contexts incorporates only a small scenario. During the workshop, the need for detailed explanations and the collection of parameters by the moderator became apparent. Future research should therefore incorporate limitations of the current work. A tool-based, guided parameter gathering will simplify the scenario characterization during workshops. Considering real-world infrastructures provides insights regarding the applicability of the approach for complex infrastructures. Sensitivity analyses allow to demonstrate the applicability and performance of selected task realization strategies, depending on the environmental conditions. The question whether the dominance of strategies can be identified remains. Further work on task realization strategies could improve task distribution and progessing. Even more specific models, such as the pricing of third parties, can increase the reference of simulations to reality. Furthermore, the fit of contemporary IT and parameter-specified architectural frameworks can be optimized at various points so that this contribution may advance our understanding of how to best realize tasks in analytical infrastructures.

## 7. Acknowledgments

This research was partly funded by the Advanced Sensor Networks SARChI Chair program, co-hosted by the University of Pretoria (UP) and Council for

Scientific and Industrial Research (CSIR), through the National Research Foundation (NRF) of South Africa.

## 8. References

Augonnet, C., Thibault, S., Namyst, R., & Wacrenier, P. (2010). Starpu: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation: Practice and Experience*, *23*(2), 187–198.

Bender, B. & Grum, M. (2016). Entwicklung eines architekturkonzepts zum flexiblen einsatz von analytics. In *Informatik 2016, 46. Jahrestagung der Gesellschaft für Informatik, 26.-30. September 2016, Klagenfurt, Österreich*, (pp. 815–824).

Brooks, D., Tiwari, V., & Martonosi, M. (2000). Wattch: A framework for architectural-level power analysis and optimizations. *SIGARCH Comput. Archit. News*, *28*(2), 83–94.

Cevher, V., Becker, S., & Schmidt, M. (2014). Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Processing Magazine*, *31*(5), 32–43.

Davenport, T. & Harris, J. (2007). *Competing on Analytics: The New Science of Winning* (1 ed.). Harvard Business Review Press.

Donahue, G. A. (2011). *Network Warrior*. O'Reilly Media, Incorporate.

EMC-Education-Services (2015). *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*. John Wiley & Sons.

Gregor, S. & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *Management Informations Systems Quarterly*, *37*(2), 337–356.

Gronau, N., Grum, M., & Bender, B. (2016). Determining the optimal level of autonomy in cyber-physical production systems. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, (pp. 1293–1299).

Gross, D., Shortle, J. F., Thompson, J. M., & Harris, C. M. (2008). *Fundamentals of Queueing Theory* (4th ed.). New York, NY, USA: Wiley-Interscience.

Grozev, N. & Buyya, R. (2014). Inter-cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, *44*(3), 369–390.

Grum, M., Bender, B., & Alfa, A. (2017). The construction of a common objective function for analytical infrastructures. *ICE IEEE ITM*, *14*(3), 342–351.

Gupta, R. A. & Chow, M. Y. (2010). Networked control system: Overview and research trends. *IEEE Transactions on Industrial Electronics*, *57*(7), 2527–2535.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *Management Informations Systems Quarterly*, *28*(1), 75–105.

Jadeja, Y. & Modi, K. (2012). Cloud computing - concepts, architecture and challenges. In *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, (pp. 877–880).

Kopetz, H. (2011). *Real-Time Systems: Design Principles for Distributed Embedded Applications* (2 ed.). Real-Time Systems Series. United States of America: Springer Publishing Company, Incorporated.

Lopez, P. G., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., & Riviere, E. (2015). Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, *45*(5), 37–42.

Luo, L., Wu, W., Di, D., Zhang, F., Yan, Y., & Mao, Y. (2012). A resource scheduling algorithm of cloud computing based on energy efficient optimization methods. In *2012 International Green Computing Conference (IGCC)*, (pp. 1–6).

May Al-Roomi, Shaikha Al-Ebrahim, S. B. & Ahmad, I. (2013). Cloud computing pricing models: A survey. *International Journal of Grid and Distributed Computing*, *6*(5), 93–106.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Management Informations Systems*, *24*(3), 45–78.

Pike, W., Bruce, J., Baddeley, B., Best, D., Franklin, L., May, R., Rice, D., Riensche, R., & Younkin, K. (2009). The scalable reasoning system: Lightweight visualization for distributed analytics. *Information Visualization*, *8*(1), 71–84.

Polyvyanyy, A., Ouyang, C., Barros, A., & van der Aalst, W. M. (2017). Process querying: Enabling business intelligence through query-based process analytics. *Decision Support Systems*, *100*(Supplement C), 41 – 56. Smart Business Process Management.

Ren, K., Wang, C., & Wang, Q. (2012). Security challenges for the public cloud. *IEEE Internet Computing*, *16*(1), 69–73.

Vogel-Heuser, B., Kegel, G., Bender, K., & Wucherer, K. (2009). Global information architecture for industrial automation. In *Automatisierungstechnische Praxis (atp)*.

Wong, P., He, Z., & Lo, E. (2013). Parallel analytics as a service. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, (pp. 25–36)., New York, New York, United States of America. Association for Computing Machinery.

Zikopoulos, P. & Eaton, C. (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data* (1st ed.). McGraw-Hill Osborne Media.