Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Modelling HPC Usage Costs
PeCoH Deliverable D3.1 & D3.3

Nathanael Hübbe

Work Package:	WP3
Responsible Institution:	DKRZ
Date of Submission:	June 2018

Chapter 1

Introduction

This white paper covers the two deliverables D3.1 and D3.3 of the PeCoH project. This work was supported by the German Research Foundation (DFG) under grants LU 1353/12-1, OL 241/2-1, and RI 1068/7-1. Thanks also to Hendryk Bockelmann for his valuable inputs.

High performance computing is a costly endeavour by definition, the costs for current supercomputers are generally within the range of a few millions to half a billion €. However, many users of such supercomputers do not seem to be aware of the costs that are caused by their use of these machines. This is especially true in academia: The supercomputer is purchased by a scientific institution, that then distributes the right to compute on this machine to its scientists. At best, these scientists see how many CPU-hours their jobs have run, and whether their stored data is still within their allocated quota of storage space. They generally don't see what this means in terms money, though.

This report seeks to explore possibilities to close this information gap, and to provide concrete example models for a concrete cluster system. To this end, a tool has been developed to provide an overview of a jobs cost right at the end of its output, based on whatever model the machines administrators choose to use. This tool is described, along with a companion tool that uses the same models to generate statistics over many jobs. This is used on an example job selection on the current supercomputer of the German Climate Computing Center, and the results are analysed.

Finally, weaknesses of the current implementation are identified, and paths of further meaningful extensions are outlined.

Chapter 2

Model Inputs

In order to assess, how super computer costs can be modeled, it is important to understand what measurements are actually available for modeling first. The following measurements may be available on a supercomputer running SLURM:

- Node list of each job, this implicitly includes the total count of nodes that were allocated to the job
- Run time of each job
- IPMI based measurement of consumed energy (optional)
- Amount of input data
- Amount of output data

In addition to that, it is possible to obtain the amount of data that users have stored in the files they own (cold data amount). However, that is no functionality that is included in SLURM, measuring the cold data amount requires use of other tools like `du`.

Of the data that SLURM itself provides, only the first three are really useful to our purpose. The other two, the amounts of in- and output data, may be interesting figures as well, but they have some features that make them bad inputs for a cost model:

- While I/O bandwidth is a finite resource, it is only important for handling load peaks. Normal operation will never saturate the theoretical I/O bandwidth.
- I/O bandwidth may be limited in several ways: By the hard drives, by the I/O nodes' network connections, by the network switches, and by the client nodes' network connections. Large installations are typically fully limited by the network parts. The clients network interfaces, however, are shared between I/O and inter-process communication via MPI, and the network switches are shared between nodes which are usually processing jobs from different users. Thus, the currently available I/O bandwidth depends on the amount of inter-process communication that is going on at the time, including communication by other jobs started by other users which run on other nodes.
- It is unknown how much I/O data will actually be transferred throughout the lifetime of the machine. As such, it is impossible to calculate a job's share of that number when the job finishes and has to be accounted for.

These factors turn the results of any attempt to estimate the cost of I/O bandwidth usage into arbitrary numbers. As such, we concentrated on the other three factors which provide a much simpler connection to the attached costs: The node list, the run time, and the consumed energy.

Chapter 3

Models for the Mistral

Based on the inputs outlined above, we developed four distinct cost models for the Mistral, which is the current supercomputer of the German Climate Computing Center (DKRZ). This machine, installed in two phases in 2015 and 2016, has cost roughly 40 M€, and consists of 3336 nodes that are available to the users.[1] The Mistral is projected to run for a total of five years, after which it will be replaced by its successor relatively quickly.

3.1 Units

For the following discussion, we will use the following units:

dimension	unit definition	symbol
time	one year	$1\,a$
	one month	$1\,mon$
data	one byte	$1\,B$
	one gigabyte	$1\,GB = 10^9\,B$
	one gibibyte	$1\,GiB = 2^{30}\,B$
	one terabyte	$1\,TB = 10^{12}\,B$
	one petabyte	$1\,PB = 10^{15}\,B$
money	one Euro	$1\,€$
	one kilo-Euro	$1\,k€ = 1000\,€$
	one mega-Euro	$1\,M€ = 10^6\,€$
hardware	one logical computer	$1\,node$
	one graphics card	$1\,GPU$
	one magnetic tape cartridge	$1\,tape$
	library space for $1\,tape$	$1\,slot$

Note that we make consistent and liberal use of SI unit prefixes and binary prefixes. As such, we may either use the base ten unit $1\,GB$ meaning exactly $1,000,000,000\,B$, or we may use the base two unit $1\,GiB$ meaning exactly $2^9\,B = 1073741824\,B$. These two are two different units which convert as $1\,GiB = 1.073741824\,GB$. We use each as appropriate, which means that RAM is measured in GiB while persistent storage is measured in GB. Likewise, we may use SI unit prefixes for base units that are not

[1] The webpage at `https://www.dkrz.de/up/systems/mistral/configuration` currently lists 3348 nodes. It appears that this page has changed slightly several times including the time when we defined the cost models. As the deviations are less than 1% of the total node count, and our models are supposed to be exemplary, not overly precise, we have not updated our calculations to be up-to-date with the mistral configuration.

normally used with these prefixes. This is most prominent with money, where we say $40\,M€$ instead of the unnecessarily lengthy $40,000,000\,€$.

3.2 Simple

This model simply takes the costs of the machine, and divides them by the total amount of node hours provided by that machine to derive the basic rate at which use of all nodes is modeled. For the Mistral, this yields a basic rate of:

$$\frac{40\,M€}{5\,a \cdot 3336\,node} = 2398\frac{€}{a \cdot node}$$

3.3 Extras

The Simple cost model treated all nodes equally, ignoring the fact that they are not: Among the 3336 Mistral nodes, there are 132 nodes that contain extra hardware. This extra hardware ranges from 64 GiB of extra memory to 960 GiB extra memory plus two graphics cards. While there are only two nodes with the full extra hardware, the value of the extra hardware in these two nodes exceeds the value of a normal node significantly. The idea of the Extras model is to differentiate between the different node configurations.

To do this, the Extras model uses three classes of node groups:

1. Basic node type: Only one group which contains all nodes.
2. Extra memory: Four optional groups for nodes with 64 GiB, 192 GiB, 448 GiB, and 960 GiB extra memory.
3. GPU: One optional group for nodes with two graphics cards.

To estimate the value of the extra hardware, we looked at current hardware prices. We found these to be around $10\frac{€}{GiB}$ for memory and $5000\frac{€}{GPU}$ for the graphics cards in use by the Mistral. For the purposes of our model, we multiplied these market prices by a factor of two to account for all other costs connected to them (including, but not limited to installation, energy consumption, and additional infrastructure costs). Thus, we calculate costs of $20\frac{€}{GiB}$ and $10\frac{k€}{GPU}$ for the additional hardware.

Since we are now accounting additional costs for specific nodes, we need to reduce the costs for a generic node without extras accordingly, so that the entire costs for the machine add up to the 40 M€ that the German Climate Computing Center payed for the Mistral. The total amount of extra hardware that is built into the Mistral is 52992 GiB of memory, and 42 graphics cards. So, with the costs estimated above, the total costs of the extra hardware are:

$$52992GiB \cdot 20\frac{€}{GiB} + 42GPU \cdot 10\frac{k€}{GPU} = 1.48\,M€$$

Thus, had the Mistral been purchased without the extra hardware, it should have cost around $40\,M€ - 1.48\,M€ = 38.52\,M€$. From this figure, we can now derive the basic node costs as in the Simple model:

$$\frac{38.52\,M€}{5\,a \cdot 3336\,node} = 2309\frac{€}{a \cdot node}$$

This calculation results in the following cost rates:

- Basic node costs (3336 nodes): $2309 \frac{€}{a \cdot node}$
- 64 GiB extra memory (380 nodes): $\frac{64\,GiB}{5\,a \cdot node} \cdot 20 \frac{€}{GiB} = 256 \frac{€}{a \cdot node}$
- 192 GiB extra memory (123 nodes): $\frac{192\,GiB}{5\,a \cdot node} \cdot 20 \frac{€}{GiB} = 768 \frac{€}{a \cdot node}$
- 448 GiB extra memory (7 nodes): $\frac{448\,GiB}{5\,a \cdot node} \cdot 20 \frac{€}{GiB} = 1792 \frac{€}{a \cdot node}$
- 960 GiB extra memory (2 nodes): $\frac{960\,GiB}{5\,a \cdot node} \cdot 20 \frac{€}{GiB} = 3840 \frac{€}{a \cdot node}$
- 2 graphics cards (21 nodes): $\frac{2\,GPU}{5\,a \cdot node} \cdot 10 \frac{k€}{GPU} = 4000 \frac{€}{a \cdot node}$

Note that these six node groups are not all mutually exclusive. For example, the two most expensive nodes are part of the first group that contains all nodes, they are part of the 960 GiB extra memory group, and they are part of the graphics cards group. Thus, these two nodes are estimated to cost $2309 \frac{€}{a \cdot node} + 3840 \frac{€}{a \cdot node} + 4000 \frac{€}{a \cdot node} = 10149 \frac{€}{a \cdot node}$ each.

3.4 Full

The goal of the Full cost model is, to give an overview of the different ways the money is spent, and to include the running costs into the calculation. To assess these costs, we divide the Mistral into three subsystems:

- Compute (60% of procurement): These are the 3336 user-visible nodes which are used to run user jobs, either interactively or via the job-scheduler.
- Online Storage (25% of procurement): The lustre file system that contains spinning disks providing 54 PB of storage space.
- Offline Storage (15% of procurement): Tape-archives that are used for long-term storage of large data volumes.

For all these parts, we assumed an infrastructure overhead of 30% on top of the pure hardware costs. Since the infrastructure overhead is included in the procurement costs, the pure hardware costs are $\frac{40\,M€}{1.3} = 30.8\,M€$. Thus, we arrive at the following basic costs:

- Compute: $60\% \cdot 30.8\,M€ = 18.5\,M€$
- Online Storage: $25\% \cdot 30.8\,M€ = 7.7\,M€$
- Offline Storage: $15\% \cdot 30.8\,M€ = 4.6\,M€$
- Infrastructure: $30\% \cdot 30.8\,M€ = 9.2\,M€$

In addition to these procurement costs, we also account for the running costs of the machine: The annual budget of the German Climate Computing Center is roughly 8.5 M€, of which roughly 2 M€ are spent to pay the energy bills. Thus, we add two more rates which we distribute evenly across all nodes:

- Power: $2 \frac{M€}{a}$
- Other: $6.5 \frac{M€}{a}$

In this model, our goal was not to account for all extra hardware precisely, so we just used two classes of compute nodes: 132 fat nodes with 256 GiB of main memory or more extras, and $3336 - 132 = 3204$ normal compute nodes. Compared to the Extras model, assuming 128 GiB main memory normal, the extra hardware costs are reduced to:

$$(123 \cdot 128\,GiB + 7 \cdot 384\,GiB + 2 \cdot 896\,GiB)20\frac{€}{GiB} + 42GPU \cdot 10\frac{k€}{GPU} = 824\,k€$$

Thus, we get the following base costs for the two node classes: $\frac{18.5\,M€ - 824\,k€}{3336\,node} = 5.3\frac{k€}{node}$ for a standard node, and $5.3\frac{k€}{node} + 824\frac{k€}{132\,node} = 11.5\frac{k€}{node}$ for the fat nodes.

For a first, rough model, we distributed the storage system, infrastructure, and running costs evenly across all nodes. Thus, we get the following rates:

- Basic node costs (3336 nodes):
  - Procurement: $\frac{5.3\,k€}{node \cdot 5\,a} = 1060\frac{€}{a \cdot node}$
  - Online Storage: $\frac{7.7\,M€}{3336node \cdot 5\,a} = 462\frac{€}{a \cdot node}$
  - Offline Storage: $\frac{4.6\,M€}{3336node \cdot 5\,a} = 276\frac{€}{a \cdot node}$
  - Infrastructure: $\frac{9.2\,M€}{3336node \cdot 5\,a} = 552\frac{€}{a \cdot node}$
  - Power: $\frac{2\,M€}{3336node \cdot a} = 600\frac{€}{a \cdot node}$
  - Other: $\frac{6.5\,M€}{3336node \cdot a} = 1948\frac{€}{a \cdot node}$
- Fat node extra costs (132 nodes): $\frac{1}{5\,a}(11.5\frac{k€}{node} - 5.3\frac{k€}{node}) = 1240\frac{€}{a \cdot node}$

3.5 Partitioned

In the Full cost model, the costs for storage were distributed evenly across the compute time that the entire system provides. This is essentially wrong: The costs for storage are mostly driven by the amount of data that needs to be held in storage, not by how much a user computes on the machine, or even how much I/O they produce with their jobs.

The Partitioned cost model tries to fix this by taking the storage costs out of the equation. These costs need to be accounted for by determining the amount of stored data a user has.

However, this change is complicated by two issues: First, the running costs and the infrastructure overhead apply to the entire system, and second, tapes for the offline storage are purchased from the German Climate Computing Center budget.

Thus, we need to first take the costs of the tape procurements out of the general German Climate Computing Center budget to add them to the offline storage only, and then divide the remaining running and infrastructure costs among the three subsystems. To estimate the costs for the tapes, we apply the following assumptions:

- The German Climate Computing Center has tape libraries with 77000 slots for tapes.
- The tape slots are always full, storage space is expanded by replacing old tapes with newer ones incrementally.
- Tapes are replaced every five years, and are LTO-tapes.
- LTO-tape generations last about 2.5 years, and provide a capacity increase of 2x over the preceding generation.

- A tape costs around 25 € when it is purchased. Currently, these are LTO-6 tapes that store 2.5 TB of data.

With these assumptions we can conclude that the German Climate Computing Center spends $\frac{77000\,tapes}{5a} \cdot 25\frac{€}{tape} = 385\frac{k€}{a}$ on tapes on average. Thus, we reduce the German Climate Computing Center budget to $6.5\frac{M€}{a} - 385\frac{k€}{a} = 6.1\frac{M€}{a}$.

With this adjusted German Climate Computing Center budget, we can proceed to distribute the running and infrastructure costs across the different systems. We do so by using the same split percentages as for the procurement. However, we need to make an exception for the power consumption of the offline storage: Offline storage is basically tapes, which do not consume any power unless they are actively read or written, and this power is comparatively negligible. (This is in contrast to online storage, which has constant power consumption due to being built of spinning disks.) Thus, Procurement, Infrastructure, and Other budget costs are split in a $\frac{60}{100}$, $\frac{25}{100}$, $\frac{15}{100}$ fashion, while Power is split in a $\frac{60}{85}$, $\frac{25}{85}$, 0 fashion. With that, we arrive at the following numbers:

- Compute (60%):
  - Procurement: $\frac{18.5\,M€}{5a} = 3.7\frac{M€}{a}$
  - Infrastructure: $60\%\frac{9.2\,M€}{5a} = 1.1\frac{M€}{a}$
  - Power: $\frac{60}{85} \cdot 2\frac{M€}{a} = 1.4\frac{M€}{a}$
  - Other: $60\% \cdot 6.1\frac{M€}{a} = 3.7\frac{M€}{a}$

  Total: $9.9\frac{M€}{a}$
- Online Storage (25%):
  - Procurement: $\frac{7.7\,M€}{5a} = 1.5\frac{M€}{a}$
  - Infrastructure: $25\%\frac{9.2\,M€}{5a} = 460\frac{k€}{a}$
  - Power: $\frac{25}{85} \cdot 2\frac{M€}{a} = 588\frac{k€}{a}$
  - Other: $25\% \cdot 6.1\frac{M€}{a} = 1.5\frac{M€}{a}$

  Total: $4.1\frac{M€}{a}$
- Offline Storage (15%):
  - Procurement: $\frac{4.6\,M€}{5a} = 920\frac{k€}{a}$
  - Infrastructure: $15\%\frac{9.2\,M€}{5a} = 276\frac{k€}{a}$
  - Other: $15\% \cdot 6.1\frac{M€}{a} = 915\frac{k€}{a}$
  - Tapes: $385\frac{k€}{a}$

  Total: $2.5\frac{M€}{a}$

To derive the individual rates for the automatic job analysis, we used the same method of distinguishing between lean and fat nodes as with the Full model. Since the two storage subsystems fall out of the compute-time based model, the remaining rates are as follows:

- Basic node costs (3336 nodes):
  - Procurement: $\frac{5.3\,k€}{node \cdot 5\,a} = 1060\frac{€}{a \cdot node}$
  - Infrastructure: $\frac{1.1\,M€}{3336 node \cdot a} = 330\frac{€}{a \cdot node}$

  - Power: $\frac{1.4\,M€}{3336node \cdot a} = 420\frac{€}{a \cdot node}$
  - Other: $\frac{3.7\,M€}{3336node \cdot a} = 1109\frac{€}{a \cdot node}$
- Fat node extra costs (132 nodes): $\frac{1}{5\,a}(11.5\frac{k€}{node} - 5.3\frac{k€}{node}) = 1240\frac{€}{a \cdot node}$

In addition to these computation based rates, we need to define storage based rates. This can be achieved by dividing the system costs by the total storage the system provides. For the online storage, it is also necessary to introduce a utilization factor for the system: When the system is purchased, it is never fully utilized; full utilization usually only happens near the end of the life-span of the system, and is something with dire consequences for the users, as it means that their jobs may crash due to insufficient storage for their output. For simplicity, we will assume an average utilization of 50% here. Thus, we get the following rates for the online storage:

- Procurement: $\frac{1.5\,M€}{50\% \cdot 54\,PB \cdot a} = 55.56\frac{€}{TB \cdot a}$
- Infrastructure: $\frac{460\,k€}{50\% \cdot 54\,PB \cdot a} = 17.04\frac{€}{TB \cdot a}$
- Power: $\frac{588\,k€}{50\% \cdot 54\,PB \cdot a} = 21.78\frac{€}{TB \cdot a}$
- Other: $\frac{1.6\,M€}{50\% \cdot 54\,PB \cdot a} = 59.26\frac{€}{TB \cdot a}$

Offline storage is assumed to be always full, so no utilization factor is necessary. Also, offline data is usually stored for exactly ten years. This term follows from formal regulations requiring scientists to archive the data on which they base their papers. During this lifetime, the data will be copied over to new tapes once, which will happen after approximately five years to free up tape slots in the tape libraries. Since the German Climate Computing Center always skips a generation of LTO tapes, and each generation stores twice as much data as the preceding generation, the data will consume only a quarter of the initial space after its move to the new tape. As such, data that is moved into the archive now produces tape costs of $(1 + \frac{1}{4}) \cdot 25\frac{€}{tape}/2.5\frac{TB}{tape} = 12.5\frac{€}{TB}$.

In addition to the tape costs, the archived data takes up space in the tape library. We estimate these according to the 77000 tape slots that are available within the tape libraries. Thus, data that is stored for ten years causes costs in terms of library slots of $5a/2.5\frac{TB}{slot} + 5a/10TBslot = 2.5\frac{slot \cdot a}{TB}$. Thus, we get the following rates for the offline storage costs:

- Procurement: $2.5\frac{slot \cdot a}{TB} \cdot 920\frac{k€}{a}\frac{1}{77000slot} = 29.87\frac{€}{TB}$
- Infrastructure: $2.5\frac{slot \cdot a}{TB} \cdot 276\frac{k€}{a}\frac{1}{77000slot} = 8.96\frac{€}{TB}$
- Other: $2.5\frac{slot \cdot a}{TB} \cdot 915\frac{k€}{a}\frac{1}{77000slot} = 29.71\frac{€}{TB}$
- Tapes: $12.5\frac{€}{TB}$

In total, these are costs of $(29.87 + 8.96 + 29.71 + 12.50)\frac{€}{TB} = 81.04\frac{€}{TB}$.

We can compare these costs to the prices which online storage providers charge for their disk space. One of these is Amazon's S3 service with prices starting at $12.50\frac{\$}{mon \cdot TB}$ for disk storage, and $4\frac{\$}{monTB}$ for tape storage. These prices do not include data access, users of Amazon's S3 service have to pay extra to access their data. Also, taxes are not included in these prices.

Converting the German Climate Computing Center storage costs calculated above to the same units that are used to charge for Amazon S3 usage, using a currency conversion rate of $1€ = 1.18\$$, we get $153.64\frac{€}{TB \cdot a} \cdot 1.18\frac{\$}{€} \cdot \frac{a}{12mon} = 15.11\frac{\$}{mon \cdot TB}$ for online

Cost Model	Compute Time / $\frac{€}{h \cdot node}$		Storage / $\frac{€}{mon \cdot TB}$		Total / $\frac{M€}{a}$
	min	max	online	offline	
Simple	0.27	0.27	-	-	8
Extras	0.26	1.16	-	-	8
Full	0.56	0.70	-	-	16.5
Partitioned	0.33	0.47	12.80	0.68	16.5

Table 3.1: Comparison of the total costs for compute time, online, and offline storage. Data in offline storage is assumed to live for ten years, and the costs are averaged over this timespan. Archiving data for only five years would increase this figure, archiving it longer than ten years would decrease the average yearly costs.

storage, and $81.04\frac{€}{TB} \cdot \frac{1}{120 mon} \cdot 1.18\frac{\$}{€} = 0.80\frac{\$}{mon \cdot TB}$ for offline storage. Considering that our estimated costs already include data access, the German Climate Computing Center storage solution seems quite competitive.

In the case of the author, who has 15.6 GB of data stored on the Mistral's online storage, and none on the offline storage, that means that the following costs would be estimated:

- Procurement: $15.6\,GB \cdot 55.56\frac{€}{TB \cdot a} = 0.87\frac{€}{a}$
- Infrastructure: $15.6\,GB \cdot 17.04\frac{€}{TB \cdot a} = 0.27\frac{€}{a}$
- Power: $15.6\,GB \cdot 21.78\frac{€}{TB \cdot a} = 0.34\frac{€}{a}$
- Other: $15.6\,GB \cdot 59.26\frac{€}{TB \cdot a} = 0.92\frac{€}{a}$

These are total storage costs of $(0.87 + 0.27 + 0.34 + 0.92)\frac{€}{a} = 2.4\frac{€}{a}$ for the author. Obviously, other users have more data stored on the Mistral.

3.6 Model Comparison

In this section, we developed four different cost models for a single supercomputer. All of these cost models focus on a different aspect, and it is hardly possible to declare any of these models as the correct one. However, in all the details of the different models, the overview of the differences may have got lost. To address this, we show the central rates of the different models in table 3.1.

This table shows the range of costs that are associated with using a node-hour of computation time. The Simple model does not differentiate between the nodes, and thus the min and max costs are the same. The Extras model goes into significant detail, with the result that the two nodes with the most hardware extensions are estimated to cost roughly 4.5 times as much as a standard node. However, the machine only has very few nodes with extra hardware, and of these most contain just a few extra GiB RAM, so the minimum costs which are estimated for the standard nodes are not much reduced compared to the Simple model.

The Full and Partitioned models add the running costs of the German Climate Computing Center to the cost estimates, increasing the total annual costs from 8 M€ to 16.5 M€. Both only distinguish between standard and fat nodes, so the cost ranges for compute time are significantly smaller than the range of the Extras model. The Partitioned cost model calculates with the same total costs as the Full model, however, it

distributes these costs among compute time and data storage. As such, the estimated costs for compute time are significantly reduced compared to the Full model. However, as only 40% of the costs are attributed to the storage solutions, the remaining compute time costs are still higher than in the Simple model.

Chapter 4

Cost Model Description Format

In order to allow for automatic analysis of jobs according to the four different cost models above, we need a well-defined format to express these models. This format is defined as follows:

- Any # character, and all characters between the # and the next newline character or end of file are removed as comments.
- Any run of whitespace that does not break the line is equivalent to a single space.
- Empty lines and leading whitespace are ignored.
- Every non-empty line contains exactly one command.
- A command consists of the command name followed by space separated arguments which may not contain whitespace. Each command defines which arguments are expected.

The list of available commands is as follows:

- *currency <string>*
  Sets the currency to use. This is for output formatting purposes only, the tools do not perform any currency conversion. It is an error to set the currency more than once, if no `currency` command is used, the currency defaults to `"dollar"`.
- *nodes <name> <nodelist> [...]*
  Create a new set of nodes. The name is used to identify the node set in the output. The nodelist is a SLURM-style description of which nodes belong to the node set, which may contain spaces. All the following rates will be relative to this node set until the next `nodes` command is encountered.
- *rate <name> <value> <unit>*
  Define a time-based rate for the current node set. Again, the name is used to identify the rate in the output. The value is some floating point number that is multiplied by the given unit.

  The unit is of the format `<multiplicator>/<time>`. Legal values for the multiplicator are `M`, `k`, `1`, `c`, and `m` which imply a factor of 1000000, 1000, 1, 0.01, and 0.001, respectively. The time unit must be one of `a`, `mon`, `w`, `d`, `h`, `min`, or `s`, which stand for a year, month, week, day, hour, minute, or second, respectively.

- *energy-rate <name> <value> <unit>*
  This is analogue to the `rate` command, with the only difference that the time unit is replaced by the string `kWh`. These rates will be calculated relative to the consumed energy that has been recorded by SLURM. If no energy record is found for the job, no `energy-rate` will be accounted for.

The formal definitions for the four models are shown in Listings 1, 2, 3, and 4.

Listing 1 Definition of the Simple model.
Due to some change of the configuration of Mistral, the nodelists do not add up to 3336 nodes, but rather to 3339 nodes. This deviation is unfortunate, and persists throughout the other models, but is so slight that we chose not to correct it.

```
currency Euro

# This calculation assumes procurement costs of 40 million spread over five ¬
years:
# 40 MEuro / 5a / 3336node = 2398 Euro/a/node

# 3336 nodes total
nodes All mlogin[100-105] mistralpp[1-5] m[10000-11367] m[11404-11421] ¬
m[11560-11577] m[11368-11403] m[11422,11431] m[11440-11511] m[11512-11549] ¬
mg[100-111] m[20000-21115] m[21434-21577] m[21607-21769] m[21116-21385] ¬
m[21386-21417] m[21420-21433] m[21589-21590] m[21593-21606] mg[200-203] mg204 ¬
mg[205-207] mg208
    rate Procurement 2398 1/a
```

Listing 2 Definition of the Extras model.

```
currency Euro

# This calculation assumes procurement costs of 40 million spread over five ¬
years.
# The extra costs are estimated by current hardware market prices, multiplied ¬
by a factor of two to account for maintenance, energy consumption, ¬
infrastructure, etc.,
# and their total is deducted from the total procurement costs to compute the ¬
price for a node without extras.
# The assumed market prices are:
#     memory: 10 Euro/GiB
#     GPU: 5 kEuro/GPU
#
# total cost = 40 MEuro / 5a = 8 MEuro/a
# extra costs memory = (380*64 GiB + 123*192 GiB + 7*448 GiB + 2*960 GiB) * ¬
2*10 Euro/GiB / 5a = 52992 GiB * 2*10 Euro/GiB / 5a = 211968 Euro/a
# extra costs GPU = 21 node * 2 GPU/node * 2*5 kEuro/GPU / 5a = 21 * 4 ¬
kEuro/a = 84 kEuro/a
# base cost = total cost - extra costs memory - extra costs GPU = 8 MEuro/a - ¬
211968 Euro/a - 84 kEuro/a = 7704032 Euro/a
# base cost / 3336 nodes = 2309 Euro/a

# 3336 nodes total
nodes BasicNode mlogin[100-105] mistralpp[1-5] m[10000-11367] m[11404-11421] ¬
m[11560-11577] m[11368-11403] m[11422,11431] m[11440-11511] m[11512-11549] ¬
mg[100-111] m[20000-21115] m[21434-21577] m[21607-21769] m[21116-21385] ¬
m[21386-21417] m[21420-21433] m[21589-21590] m[21593-21606] mg[200-203] mg204 ¬
mg[205-207] mg208
    rate Procurement 2309 1/a

# 380 nodes total
nodes ExtraMemory-64GB m[11368-11403] m[11422,11431] m[11440-11511] ¬
m[21116-21385]
    rate Procurement 256 1/a

# 123 nodes total
nodes ExtraMemory-192GB mlogin[100-105] mistralpp[1-5] m[21386-21417] ¬
m[21420-21433] m[21589-21590] m[21593-21606] m[11512-11549] mg[100-111]
    rate Procurement 768 1/a

# 7 nodes total
nodes ExtraMemory-448GB mg[200-203] mg[205-207]
    rate Procurement 1792 1/a

# 2 nodes total
nodes ExtraMemory-960GB mg[204,208]
    rate Procurement 3840 1/a

# 21 nodes total
nodes GPU mg[100-111] mg[200-208]
    rate Procurement 4000 1/a
```

Listing 3 Definition of the Full model.

```
currency Euro

# This model is based on the following assumptions:
# Hardware + Infrastructure = 40 MEuro
# Infrastructure = 30%*Hardware
# => Hardware = 30.8 MEuro
# => Infrastructure = 9.2 MEuro
#
# Compute = 60%*Hardware = 18.5 MEuro
# Disks = 25%*Hardware = 7.7 MEuro
# Tapes = 15%*Hardware = 4.6 MEuro
#
# The annual budget of the computing center:
# Power = 2 MEuro/a
# DKRZ = 6.5 MEuro/a
#
# Nodes with less than 256 GiB memory are normal, other nodes are fat (132 ¬
nodes).
# Extra costs for fat nodes as in the extras model:
# Extras = (123*128 GiB + 7*384 GiB + 2*896 GiB)20 Euro/GiB + 42 GPU * 10 ¬
kEuro/GPU = 824 kEuro
# Lean Node = (Compute - Extras)/3336 node = 5.3 kEuro/node
# Fat Node = Lean Node + Extras/132 node = 11.5 kEuro
#
# Disks, Tapes, Infrastructure, and Running Costs are distributed evenly ¬
across all nodes.

# 3336 nodes total
nodes BasicNode mlogin[100-105] mistralpp[1-5] m[10000-11367] m[11404-11421] ¬
m[11560-11577] m[11368-11403] m[11422,11431] m[11440-11511] m[11512-11549] ¬
mg[100-111] m[20000-21115] m[21434-21577] m[21607-21769] m[21116-21385] ¬
m[21386-21417] m[21420-21433] m[21589-21590] m[21593-21606] mg[200-203] mg204 ¬
mg[205-207] mg208
    rate Procurement 1060 1/a   #5.3 kEuro / 5a
    rate Disks 462 1/a  #7.7 MEuro / 3336 nodes / 5a
    rate Tapes 276 1/a  #4.6 MEuro / 3336 nodes / 5a
    rate Infrastructure 552 1/a #9.2 MEuro / 3336 nodes / 5a
    rate Power 600 1/a  #2 MEuro / 3336 nodes
    rate DKRZ 1948 1/a  #6.5 MEuro / 3336 nodes

# 132 nodes total
nodes FatNodeExtra mlogin[100-105] mistralpp[1-5] m[21386-21417] ¬
m[21420-21433] m[21589-21590] m[21593-21606] m[11512-11549] mg[100-111] ¬
mg[200-203] mg[205-207] mg[204,208]
    rate Procurement 1240 1/a   #(11.5 kEuro - 5.3 kEuro) / 5a
```

Listing 4 Definition of the Partitioned model.

```
currency Euro

# This model is based on the Full model,
# however, it removes the costs for storage along with their procentual share ¬
of the non-hardware costs,
# assuming that they are accounted for by analysing the amount of stored data.
#
# This model is based on the following assumptions:
# Hardware + Infrastructure = 40 MEuro
# Infrastructure = 30%*Hardware
# => Hardware = 30.8 MEuro/5a = 6.16 MEuro/a
# => Infrastructure = 9.2 MEuro/5a = 1.84 MEuro/a
#
# The annual budget of the computing center, total 8.5 MEuro/a:
# Power = 2 MEuro/a, distributed 60/85, 25/85, 0 to Compute, OnlineStorage, ¬
and OfflineStorage, respectively.
# Tapes = 77000 tapes / 5a * 25 Euro/tape = 385 kEuro/a
# DKRZ = 8.5 MEuro/a - 2 MEuro/a - 385 kEuro/a = 6.1 MEuro/a, distributed 60%, ¬
25%, 15% to Compute, OnlineStorage, and OfflineStorage, respectively.
#
# What we account for in this model:
# Compute = 60%*Hardware = 3.7 MEuro/a
# ComputeInfrastructure = 60%*Infrastructure = 1.1 MEuro/a
# ComputePower = 60/85*Power = 1.4 MEuro/a
# ComputeDKRZ = 60%*DKRZ = 3.7 MEuro/a
#
# What we do not account for:
# Disks = 25%*(Hardware + Infrastructure + DKRZ) + 25/85*Power = 4.1 MEuro/a
# Tapes = 15%*(Hardware + Infrastructure + DKRZ) + Tapes = 2.5 MEuro/a
#
# Nodes with less than 256 GiB memory are normal, other nodes are fat (132 ¬
nodes).
# Extra costs for fat nodes as in the extras model:
# Extras = (123*128 GiB + 7*384 GiB + 2*896 GiB)20 Euro/GiB + 42 GPU * 10 ¬
kEuro/GPU = 824 kEuro
# Lean Node = (Compute - Extras/5a)/3336 node = 1060 Euro/node/a
# Fat Node = Lean Node + Extras/5a/132 node = 2.3 kEuro/node/a

# 3336 nodes total
nodes BasicNode mlogin[100-105] mistralpp[1-5] m[10000-11367] m[11404-11421] ¬
m[11560-11577] m[11368-11403] m[11422,11431] m[11440-11511] m[11512-11549] ¬
mg[100-111] m[20000-21115] m[21434-21577] m[21607-21769] m[21116-21385] ¬
m[21386-21417] m[21420-21433] m[21589-21590] m[21593-21606] mg[200-203] mg204 ¬
mg[205-207] mg208
    rate ComputeProcurement 1060 1/a
    rate ComputeInfrastructure 330 1/a  #1.1 MEuro/a / 3336 nodes
    rate ComputePower 420 1/a   #1.4 MEuro/a / 3336 nodes
    rate ComputeDKRZ 1109 1/a   #3.7 MEuro/a / 3336 nodes

# 132 nodes total
nodes FatNodeExtra mlogin[100-105] mistralpp[1-5] m[21386-21417] ¬
m[21420-21433] m[21589-21590] m[21593-21606] m[11512-11549] mg[100-111] ¬
mg[200-203] mg[205-207] mg[204,208]
    rate ComputeProcurement 1240 1/a    #2.3 kEuro/node/a - 1060 Euro/node/a
```

Chapter 5

Analysis Tools

To actually apply the four models outlined above to SLURM jobs, we have developed two tools: `job-cost-meter.sh` and `batch-cost-analyser.sh`.

5.1 `job-cost-meter.sh`

The first of the two tools is a shell script that is supposed to be run from a job-epilogue script for SLURM. This script reads in a model description from a file and uses that cost model to generate a cost estimate for the current job. The result of this analysis is appended to the file containing the `stdout` of the current job to provide feedback to the user who started the job. This script can also be used for post-mortem analysis, writing its output directly to `stdout`, however, it is not suitable for performing statistical analyses.

Listing 5 Output of the single job analysis script for one of the author's jobs, using the Full cost model.

```
====================================
job cost estimate
====================================

(10 nodes total)
BasicNode (10 nodes):
    Procurement:            0.09 Euro
    Disks:                  0.04 Euro
    Tapes:                  0.02 Euro
    Infrastructure:         0.05 Euro
    Power:                  0.05 Euro
    DKRZ:                   0.17 Euro
-----------------
total:                      0.43 Euro

====================================
```

The `job-cost-meter.sh` script supports three different verbosity levels: Full output (`--verbose`) provides a job cost estimate section formatted as the examples in the Listings 5 and 6, the switch `--json` provides the same information but formats it in JSON for easy tool consumption. Short output (`--short`) produces only a single line with the total (`"job cost estimate: 0.43 Euro"`), while the quiet mode (`--quiet`) strips all formatting to produce only the raw decimal number (`"0.43"`). This last mode is meant to be

Listing 6 Output of the single job analysis script for the same job as Listing 5, but this time using the Partitioned cost model.

```
==========================================
job cost estimate
==========================================

(10 nodes total)
BasicNode (10 nodes):
    ComputeProcurement:            0.09 Euro
    ComputeInfrastructure:         0.03 Euro
    ComputePower:                  0.04 Euro
    ComputeDKRZ:                   0.10 Euro
-----------------
total:                             0.26 Euro

==========================================
```

used by scripts, allowing them to use the total without needing to parse any output.

Post mortem analysis is facilitated by the `--job` and `--nodes` options, which should always be used in tandem. The first one sets the SLURM Job ID to analyze, the second sets the node list to use to analyze the job. Specifying either `--job` or `--nodes` implies post-mortem mode in which the output of the script is not redirected from standard out to the job's output file. This may also be switched explicitly using `--post-mortem` or `--redirect`.

5.2 `batch-cost-analyser.sh`

The second tool is also a `bash` script that uses the same cost model description language as the job-epilogue script, with the difference that it is geared towards analysis of many jobs. It takes a number of job-selection parameters on the command line, which are passed through to an `sacct` invocation to produce a list of jobs that are to be analyzed. This script calculates all the different costs that are associated with each job, and stores them in an internal table. It may then either dump the contents of this table to `stdout`, produce a table with statistics for each cost rate defined by the cost model, or do both. The most important part of the statistics output is a table with configurable quantiles of the data. The statistics table also includes the total sum, element count, mean value, and standard deviation. Except for the sum, these statistical figures are provided both to include or exclude entries without data. That way, it is easy to see how many jobs actually used GPU nodes, for instance.

The `batch-job-analyser.sh` hat two different output modi: One that provides a column for each rate that was defined in the cost model, and one (`--quiet`) that provides a single data column with the totals. Furthermore, it provides output of two different tables: One that lists the costs of each job separately (`--details`), and one that lists statistical information of the jobs (`--statistics`) as shown in Listings 7 and 8. If neither `--details` nor `--statistics` is given, both tables are produced. The statistics table (`--statistics`) is further customizable by specifying a quantile increment with `--increment`, which controls the widths of the quantiles that are displayed.

Finally, the script allows the user to exclude very long running jobs with `--max-runtime`, a feature that was necessary to avoid bias from a ghost-job in the SLURM job database on mistral: The job, though long dead, was recorded to be run-

Listing 7 Left part of the statistics table produced for some of the author's jobs, using the simple cost model.

```
 statistics  ||               Size            Runtime              Energy  |
=============||=======================================================|
         0%  ||   1 nodes (  0.1%)  0.0000 h (  0.0%)                    |
        25%  ||   1 nodes ( 13.9%)  0.0016 h (  4.3%)      474 J (  0.5%)  |
        50%  ||   2 nodes ( 31.1%)  0.0022 h ( 10.9%)      996 J (  2.0%)  |
        75%  ||   2 nodes ( 58.8%)  0.0052 h ( 20.2%)     5332 J (  5.9%)  |
       100%  ||  10 nodes (100.0%)  0.1491 h (100.0%)  1462013 J (100.0%)  |
-------------||-------------------------------------------------------|
        sum  ||          854 nodes           3.5058 h           5324044 J  |
      count  ||                475                475                449  |
total count  ||                475                475                475  |
       mean  ||            2 nodes           0.0074 h             11858 J  |
    std-dev  ||            1 nodes           0.0100 h             73865 J  |
 total mean  ||            2 nodes           0.0074 h             11209 J  |
  total dev  ||            1 nodes           0.0162 h             72072 J  |
```

Listing 8 Right part of the statistics table produced for some of the author's jobs, using the simple cost model.

```
|                All         Procurement  ||               Total
|=======================================||====================
|   1 nodes (  0.1%)  0.00 Euro (  0.0%)  ||  0.00 Euro (  0.0%)
|   1 nodes ( 13.9%)  0.00 Euro (  2.5%)  ||  0.00 Euro (  2.5%)
|   2 nodes ( 31.1%)  0.00 Euro (  6.3%)  ||  0.00 Euro (  6.3%)
|   2 nodes ( 58.8%)  0.00 Euro ( 13.2%)  ||  0.00 Euro ( 13.2%)
|  10 nodes (100.0%)  0.41 Euro (100.0%)  ||  0.41 Euro (100.0%)
|---------------------------------------||--------------------
|          854 nodes           2.32 Euro  ||           2.32 Euro
|                475                475  ||                475
|                475                475  ||                475
|            2 nodes           0.00 Euro  ||           0.00 Euro
|            1 nodes           0.02 Euro  ||           0.02 Euro
|            2 nodes           0.00 Euro  ||           0.00 Euro
|            1 nodes           0.02 Euro  ||           0.02 Euro
```

ning for over half a year, a time span that was impossible to be true both due to mistral's max runtime policies, and due to the fact that there was a power outage within the recorded time span. With its enormous total run time, the ghost-job would have skewed all the total statistics.

5.3 Security Considerations

The `job-cost-meter.sh` script is designed to be run as a SLURM epilogue script. That means that it runs as root. Which in turn means that security had to be taken seriously. Unfortunately, SLURM does not really make it easy to write epilogue scripts that don't introduce security vulnerabilities into an HPC system.

To be precise, SLURM does not give epilogue scripts access to the job's `stdout/stderr` file paths in a robust way. To the knowledge of the author, the only available method of retrieving these paths is to parse the output of `scontrol show job`, which does not properly escape its controlling characters in the user-controlled strings that it includes. This makes it easy for malicious users to fool any job epilogue into writing to any file. For instance, a job could be submitted with its `stdout` connected to a file called `"evil space"`. If there were also a symbolic link `evil` present that points to `"/etc/passwd"`, a trusting job epilogue script will write to that file with root privileges.

To avoid these problems, the `job-cost-meter.sh` has been written to rely on a patched version of SLURM instead. This patch sequence introduces an additional feature to the `scontrol show` tool that provides the values of the individual fields. So, `"$(scontrol show job-stdout $SLURM_JOB_ID)"` will produce exactly that path, including all spaces, newlines, unicode characters, tabs, dots, whatever. This patch sequence has been submitted as a contribution to the SLURM developers in bug report 4086 (`https://bugs.schedmd.com/show_bug.cgi?id=4086`). Unfortunately it has not been merged, nor has the alternative of sanitizing user-supplied paths been implemented to date (2018-08-13)

Anyway, the failure to handle user-supplied strings correctly is just part of the problem. A second problem exists in the fact, that the epilogue's accesses to the `stdout` file of the job is a distinct file operation from SLURMs own use of that same file. This means that there is fundamentally no guarantee that the two operation actually work on the same file. For instance, a job could just delete its own output file and replace it with symbolic link to `/etc/passwd`. Even if that were not possible for some reason (sanitization, or some other mechanism), a user could still race the job's execution with an external script that does exactly this replacement.

Thus, the only sound way to actually perform output on behalf of the user in a job epilogue script, is to actually become that user first. I.e. to drop privileges before any output to the file system takes place. This approach is implemented in `job-cost-meter.sh`, so the `scontrol show job` parsing problem is actually a minor concern now. (Users can only trick the epilogue to do whatever they are allowed to do anyway.) Nevertheless the current version of the script still requires the patch sequence, as there has been no move to supply epilogue scripts with robust paths, yet. Once SLURM does provide something in that direction, we are likely to revisit the `job-cost-meter.sh` script. Until that happens, the patches have to be used.

Chapter 6

Statistical Results

To assess the value of the different cost models, we applied them to all the jobs run on the mistral on the tenth of September 2017. For this, the `batch-cost-analyser.sh` script was started with

```
./batch-cost-analyser.sh -i 1 -s <cost-model-config-file> ¬
-S 2017-09-10T00:00 -E 2017-09-11T00:00 -a
```

These options set the quantile interval to 1% (`-i 1`), and restrict output to the statistics table only (`-s`). The options after the cost model configuration file path are passed through to `sacct` for job selection. In this case, they select all jobs by all users (`-a`) which were active during the time interval from `2017-09-10T00:00` to `2017-09-11T00:00`. Unfortunately, this includes any job started on the preceding day that was finished after `2017-09-10T00:00`, as well as any job started before `2017-09-11T00:00` that finished some time later. As such, the selection is not representative, it over-represents long running jobs.

As shown in Listing 7, the output of `batch-cost-analyser.sh` starts with three columns providing the statistics of its inputs. The job size distribution for the selected jobs is shown in Figure 6.1, the distribution of the run time of the same jobs is shown in Figure 6.2.

Both figures are shown with a logarithmic y-axis to account for the many orders of magnitude their data spans. This is especially pronounced with the run time graph, which shows that the longest running jobs have run longer than four days, while a sizable fraction of jobs finishes within a few seconds. This means, that only a very small percentage of the jobs is responsible for the majority of resource allocations, which clearly shows in the cost analysis as well, which is shown in Figure 6.3.

All four cost models show the same statistical behavior, the differences being largely just a constant factor. This shows most clearly when comparing the Simple and the Extras model, which are nearly indistinguishable in the graph. As the Full model includes costs which the Simple and Extras model do not account for, it is no surprise that it estimates higher costs. The Partitioned model excludes other costs than the Simple and Extras models, yet its graph shows precisely the same qualitative behavior.

It should be noted, that the logarithmic range of the estimated costs is even wider than that of the job run times. The costs were computed to a precision of one cent, and the most costly jobs exceed a million cents. In addition, roughly a third of the job costs were rounded to zero cents, indicating that a significant part of the logarithmic range got lost due to precision. This suggests that the size and run times of the jobs are correlated: Big jobs generally run longer.

To get a better idea on which fractions of jobs account for which fraction of costs, we

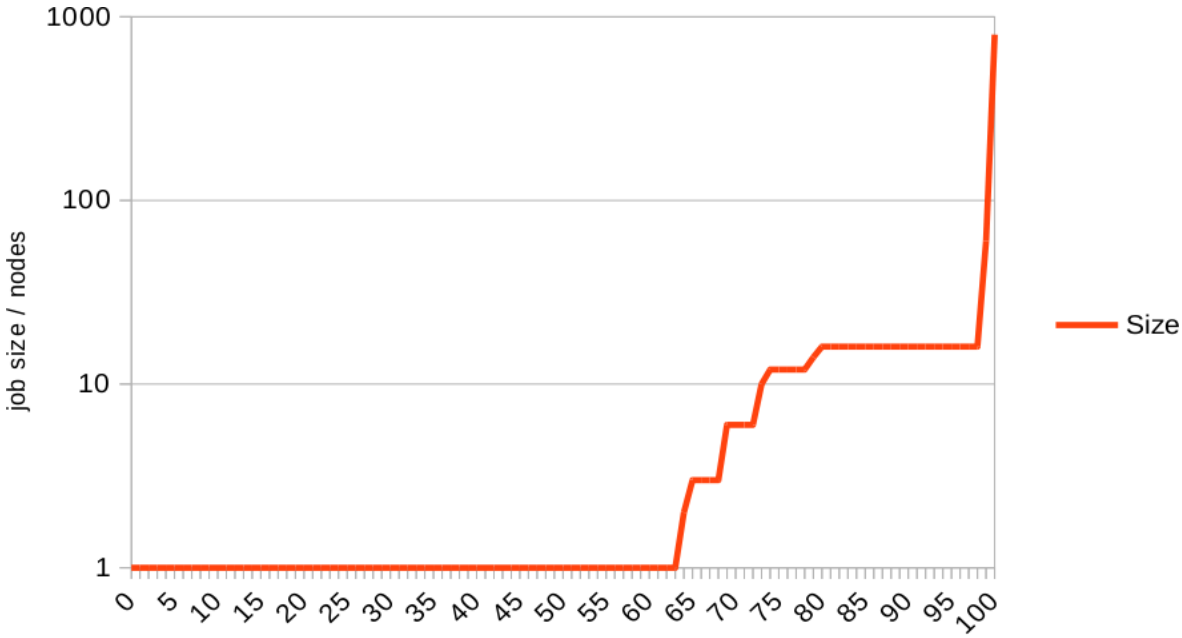

Figure 6.1: The number of nodes allocated to the selected jobs. The x-axis shows the quantile, the y-axis the size of the respective job. Note that two thirds of the jobs only allocate a single node, the plateau around the 90% quantile is at 16 nodes.

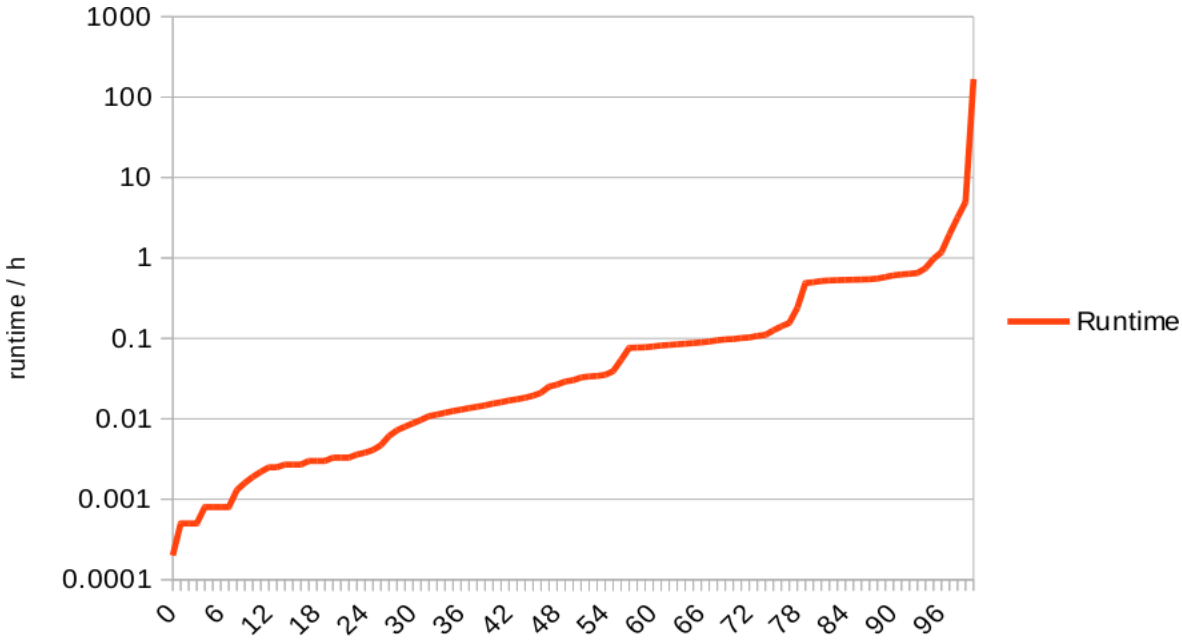

Figure 6.2: The run times of the selected jobs. The x-axis shows the quantile, the y-axis the run time of the respective job in hours. Note that the run times span almost six decimal orders of magnitude. The two jumps near the 56% and 78% quantiles indicate that users start only very few jobs with runtimes in the two intervals $(2.4min, 4.5min)$ and $(9.4min, 29.2min)$. The reasons for this are unknown to us, but we believe that psychological factors may be involved.

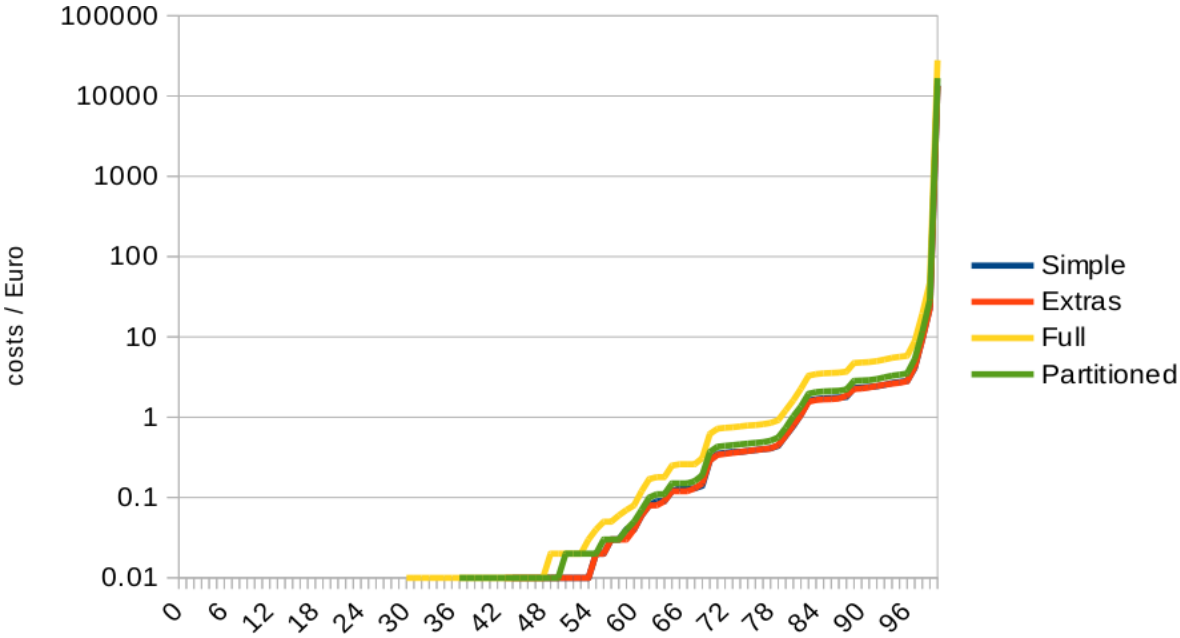

Figure 6.3: The estimated costs of the selected jobs. The x-axis shows the quantile, the y-axis the cost of the respective job as estimated by the different models. Note that the Partitioned model only shows the compute-time related costs, which is why it is lower than the Full cost model.

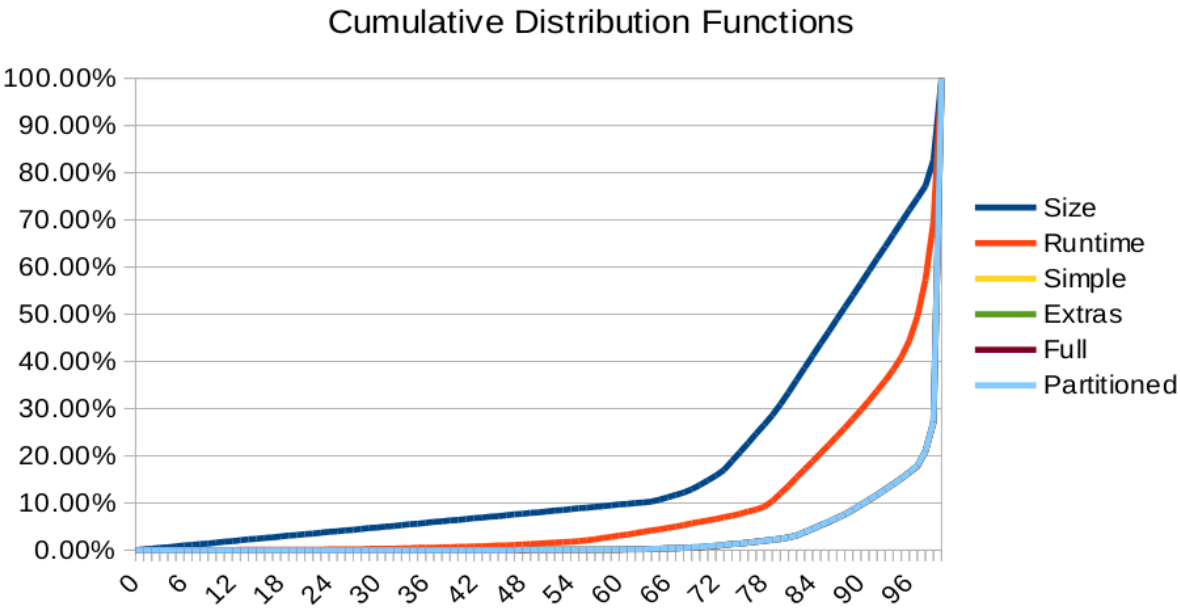

Figure 6.4: Cumulative distribution functions of the job sizes, the job run times, and the different cost estimates. All four cost models are indistinguishable from each other in this plot, they are statistically indistinguishable.

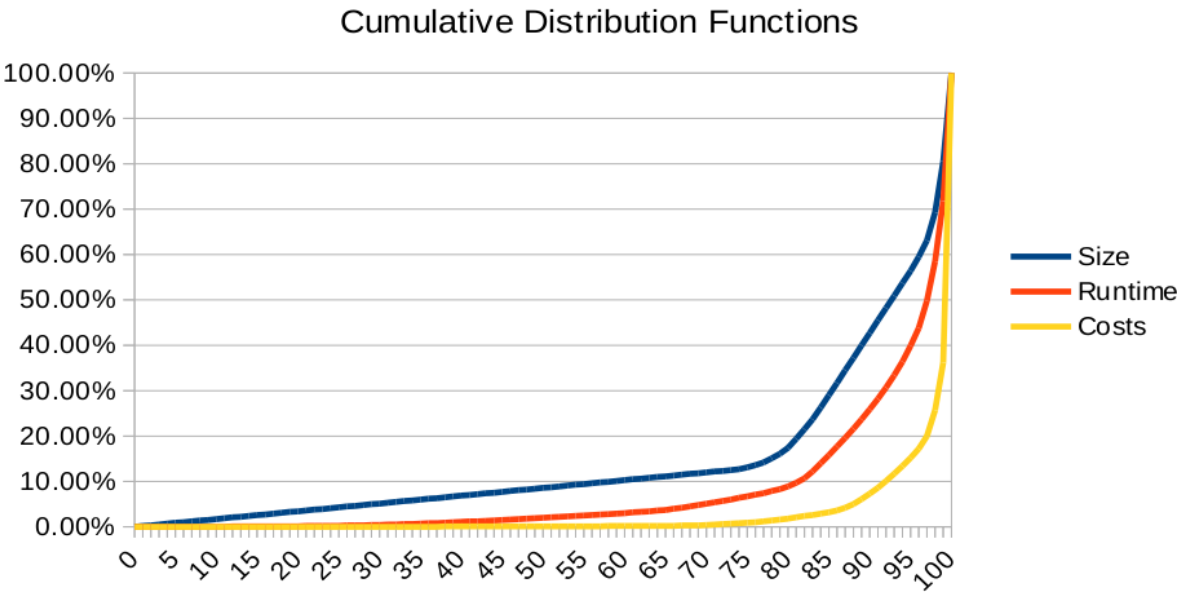

Figure 6.5: Cumulative distribution functions for jobs running the week from 2017-09-10 to 2017-09-16. The skew is somewhat reduced due to the more representative data set (compared to Figure 6.5), but is still very strong.

have also plotted the cumulative distribution functions in Figure 6.4. This is basically the data in Figures 6.1, 6.2, and 6.3 integrated and normed to 100%. The resulting graphs show, that the different cost models are indeed virtually indistinguishable from their statistical behavior, but that the distribution of the costs is much more skewed than both the distributions of the job size and their run times. The skew is so extreme, that the most costly 1% of jobs accounts for more than 70% of the costs.

However, the skew is exaggerated by the non-representativeness of the job selection: We have seen above, that the job selection over represents long running jobs, which are the most costly ones. To get an impression of how much of the skew is due to this long-job-over-representation, we also ran the Simple cost model on a full weeks worth of data. The resulting cumulative distribution functions are show in Figure 6.5.

We performed the full week run only with the Simple cost model since we had performance problems with the script due to it being a bash script, and the count of jobs that are run on the Mistral is quite large, large enough that the script took many hours to analyze a weeks worth of job data. As we had seen before that the models show the same statistical behavior, we didn't deem it necessary to perform the full week run with all cost models.

The longer time span from which the jobs were selected does somewhat reduce the skew of the distribution, however, the most costly 1% of the jobs still accumulate over 60% of the costs. The distribution is for the most part dominated by the small, short running jobs that are performed by the users to test things out. Users on the Mistral are accounted for the CPU hours they consume, so they don't seem to want to waste their compute time on these kinds of jobs. The costs are driven by the tiny fraction of large, long running jobs which typically do not get started before a series of small jobs has shown that good results are to be expected from the large job.

Chapter 7

Possible Model Extensions

The current tools that were programmed within the PeCoH project rely exclusively on input data provided by the SLURM batch job manager. These include the run time, the node allocation, and the energy consumed by the job (if energy recording is activated, that is, which is not the case on the Mistral). This is very appropriate to estimate the costs for computation time. However, as seen in the derivation of the Partitioned cost model, this is insufficient to give fully adequate feedback to the user. Doing so requires additional data sources.

As was seen in the Partitioned cost model, the most important additional data source would be file system usage. Also, when it comes to archiving data for long term storage, as supported by the German Climate Computing Center, the amounts of data that are written into the archive should be available along with the time that the data is expected to remain in the archive.

Both of these additional data sources require different hooks to be implemented to provide feedback to the user: File system usage is largely an ongoing cost factor, so it should be reported repeatedly, either as a current monthly rate, or as an accumulated sum over a given time period. As such, the data source would be a script/tool like `du` that is run at specific times (like a `cron` job) to determine the current data usage of the user.

Data that is moved into an archive, on the other hand, is expected to remain there for a fixed period of time. Thus, when the data is moved into the archive, all the information is available to determine the estimated cost of archiving that data. Consequently, the natural hooking point are the tools that are used for archiving, they would simply extend their output to include the estimated costs to provide the feedback to the user.

Another significant component of a supercomputer that we have not considered in this discussion is the network. Supercomputers tend to use comparatively potent network hardware. The problem with accounting for network usage is that it is by definition a shared resource that imposes changing limits on communication throughput. A process that communicates when no other process communicates causes no costs to others. Yet, when the communication happens to delay a packet of a 10000 node job by a millisecond, this interaction may cost the large process ten node-seconds of compute time. This makes it difficult to derive cost rates from first principles, which is why we ignored this part in our investigations.

Nevertheless, it would be possible to compute an aggregated yearly throughput that is actually used by the software running on the supercomputer, and compute the costs for network throughput relative to that figure. If that is done, it needs to be taken into account, that parallel file-system I/O uses the network hardware as well, and thus is partly responsible for a supercomputers networking costs.

Chapter 8

Summary

In this deliverable, we have outlined several different ways to model super computer costs down to individual user actions like starting jobs and moving data into the archive. This has been done using the example of the Mistral, which is the current supercomputer at the German Climate Computing Center. For the job related part of the modelling, we have created scripts that are able to both give direct cost feedback to the users running the jobs, and to perform post-mortem analysis for a statistical overview of the cost distributions. Such a post-mortem analysis has been done with actual job data on the Mistral, showing an extreme accumulation of costs to only very few jobs.

In addition to that, we have also outlined ways to refine the cost models further, taking storage usage into account as well. This analysis includes the description of which additional input data is necessary for such refined cost models, and how it may be collected. Nevertheless, actually writing tools for these refined models is beyond the scope of the PeCoH project.