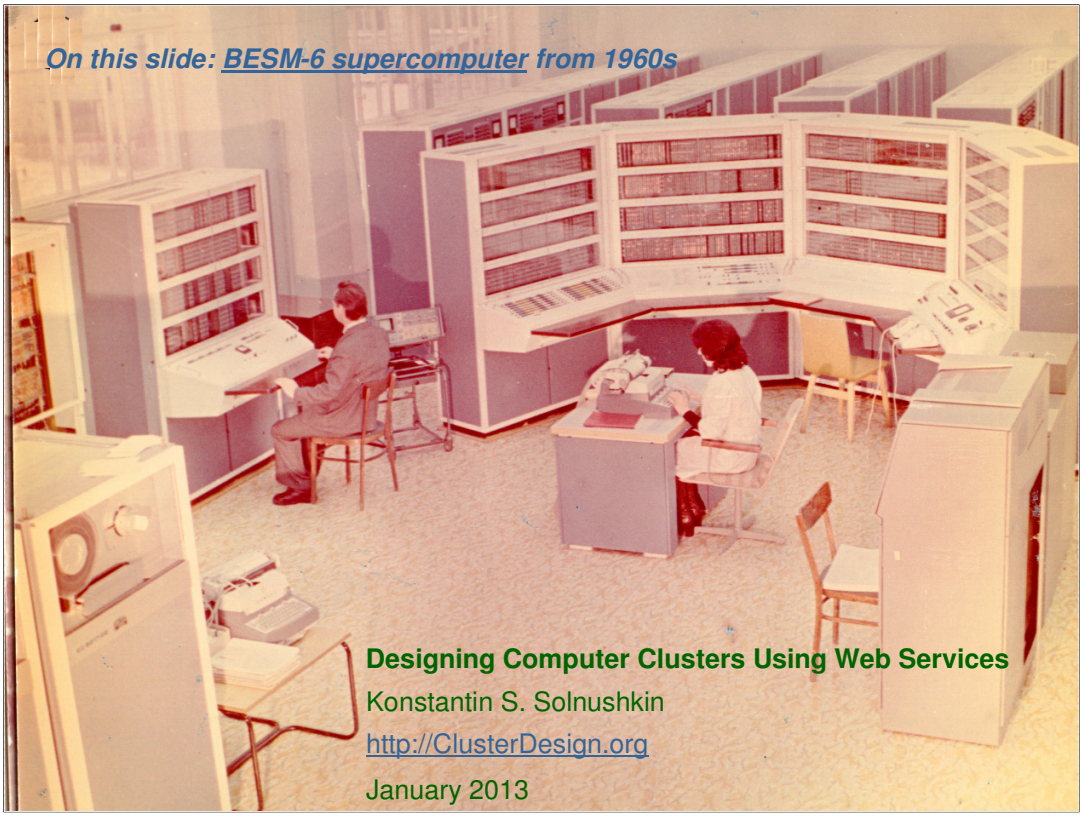


Please note these slides have helpful comments!

Presentation Outline

1. Problem statement
2. Graph-based representation of configurations
3. Performance modelling – direct and reverse
4. Design workflow
5. Web service for fat-tree and torus network design
6. Web service for ANSYS Fluent performance model
7. CAD tool for computer cluster design
8. Analysis of results
9. Future work

Hi, today we will review how automated design of computer clusters can be accomplished by querying web services – separate software modules providing required functionality, such as calculating performance, designing cluster subsystems (interconnection network, for example), and so on.

Problem Statement

$$\min_{S \in \{S\}} \frac{c_{tot}}{p}$$

c_{tot} – total cost of ownership

p – performance

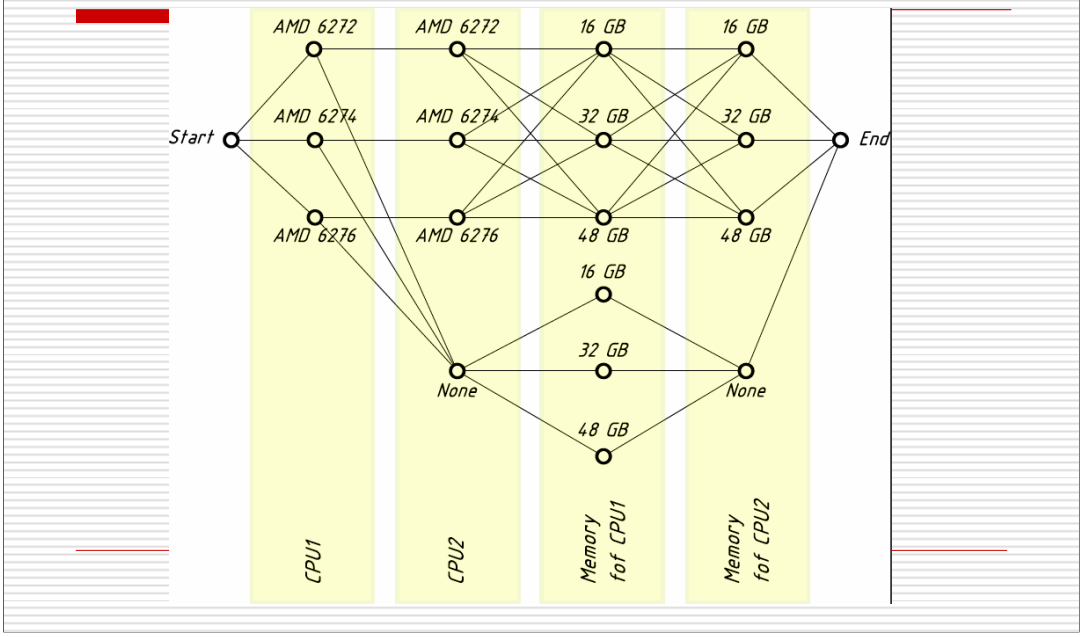
$$S = (m_{CPU}, f_{CPU}, n_{CPU}, \ldots)$$

The optimization problem is formulated as follows: find a tuple S (that precisely describes supercomputer parameters) which minimizes the objective function "total cost of ownership divided by performance of supercomputer".

One of the main challenges is representing supercomputer parameters, because a vector of fixed length (such as vector S above) is not suitable in the general case. This is especially evident when certain hardware items are not compatible, which means that corresponding combinations of parameters in vector S are forbidden.

We solve this problem by representing supercomputer configurations with a graph, and calculating parameters by traversing this graph (see next slide).

Graph-Based Representation of Configurations



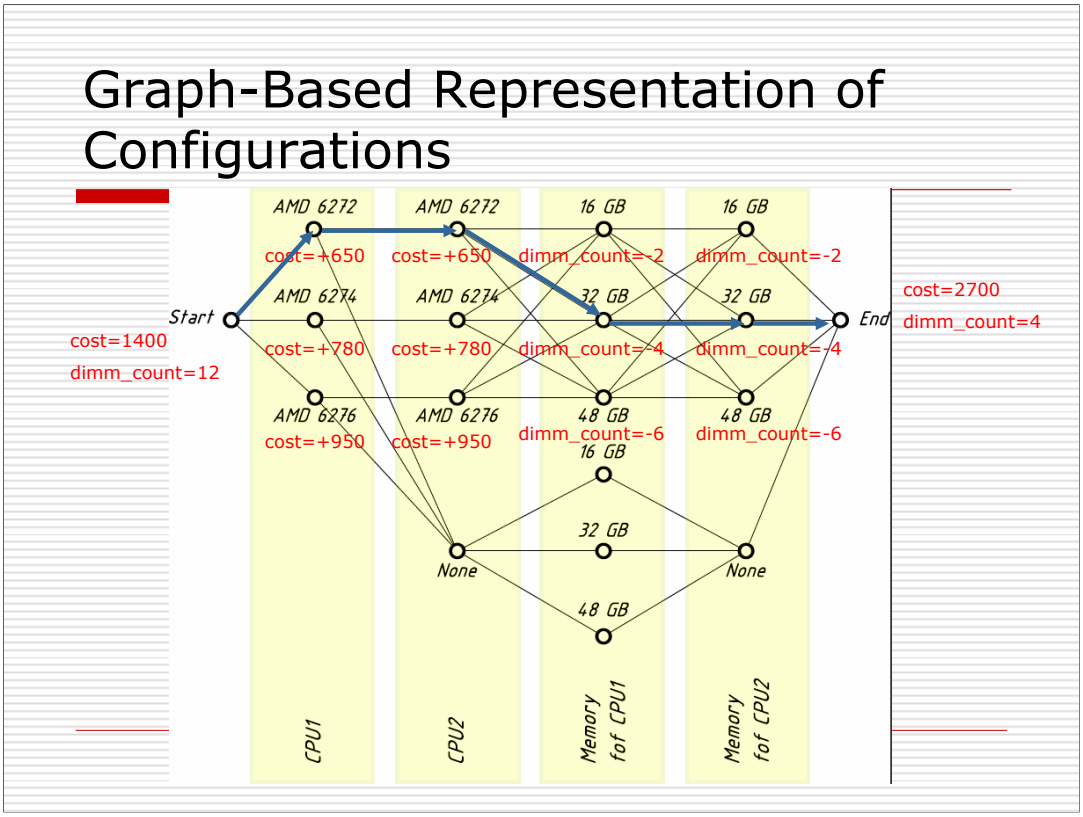
Bozhko and Tolparov proposed (see [1] below) to use undirected cyclic multipartite graphs to represent allowed combinations of elements in various technical devices, with edges representing compatibility between elements. Complex technical systems can be described using this approach, preserving precise compatibility information.

However, no provisions exist to calculate characteristics of technical systems. Therefore, we further developed this approach by assigning edges and vertices with expressions that must be evaluated. Also, we shifted from undirected graphs with cycles to directed acyclic graphs, as this provides better visual clues regarding traversing of the graph.

Here we represent a simple configuration of a cluster compute node using our approach with directed acyclic graphs. The server can have either one or two CPUs, there are three possible CPU models (AMD 6272, 6274 and 6276), and if the second CPU is installed, it must be identical to the first one.

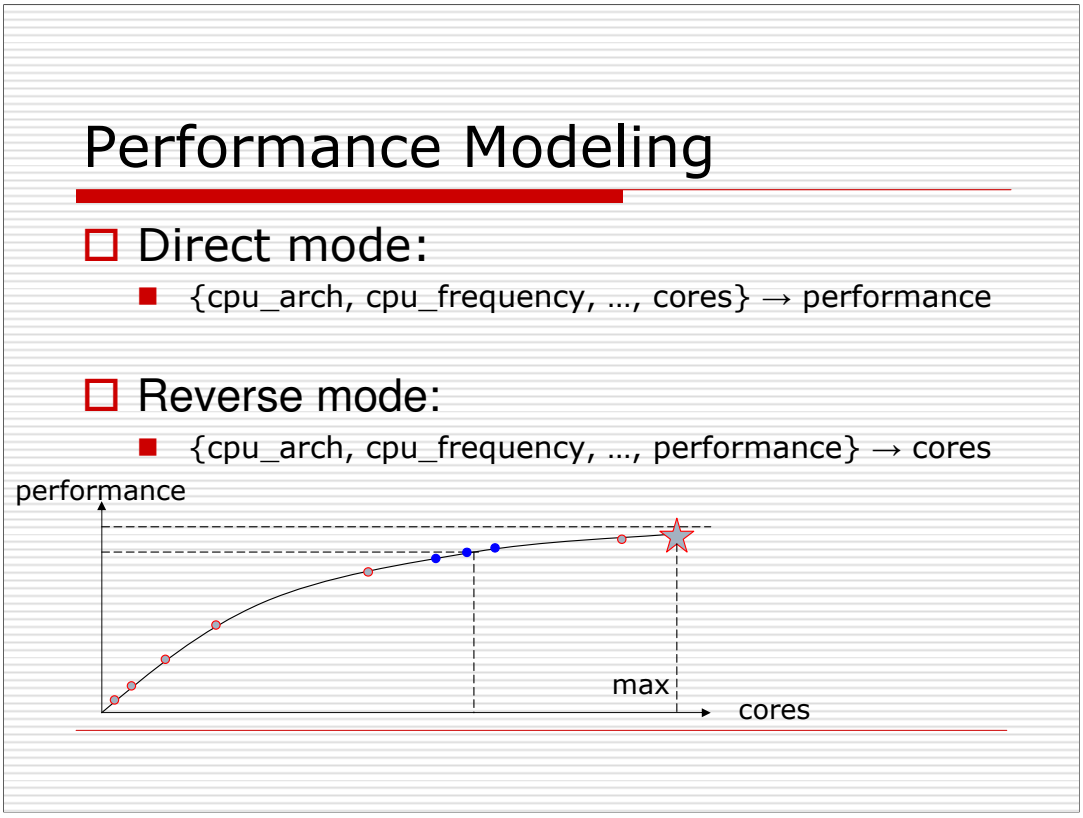
CPUs are equipped with memory separately, i.e., the first CPU can have either 16, 32 or 48 GBytes of memory attached to it, and so can the second CPU (if it is installed). Obtaining a valid configuration requires traversing a path in the graph, from “Start” to “End”. Let us prescribe (arbitrary) values and arithmetic expressions to graph vertices (see next slide).

Footnote: 1. A.N. Bozhko and A.Ch. Tolparov. Structural synthesis on elements of limited compatibility (in Russian). Science and Education, 5, 2004.



Here, we prescribe the initial cost of the server (the bare motherboard, before any components are installed) to be 1400 monetary units. The number of DIMM memory slots is set to 12. Adding CPUs increases the “cost” characteristic as per evaluated expressions (for example, “cost=+650”). Correspondingly, adding memory decreases the number of available DIMM slots (for example, “dimm_count=-4”).

As a result, traversing a path in the graph not only ensures compatibility between all components, but also simultaneously computes all necessary technical and economic characteristics. In the shown case, a particular path is highlighted in blue, and corresponding characteristics for this path are shown on the right: “cost=2700; dimm_count=4”.



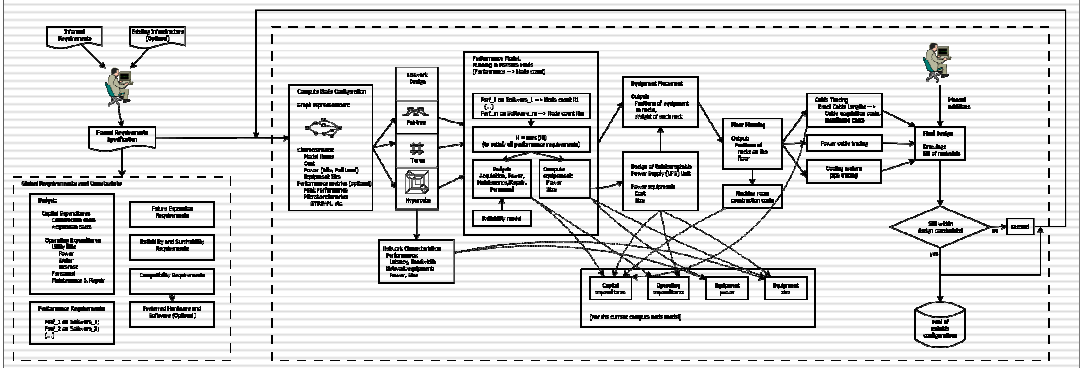
There are two performance modelling modes. Generally speaking, in the direct mode (the best known), we supply supercomputer parameters and the number of cores, and receive performance.

In the lesser known reverse mode, we supply supercomputer parameters along with desired performance, and receive the number of cores. We solve the inverse modelling task by calling the direct performance model iteratively several times. That's how it works:

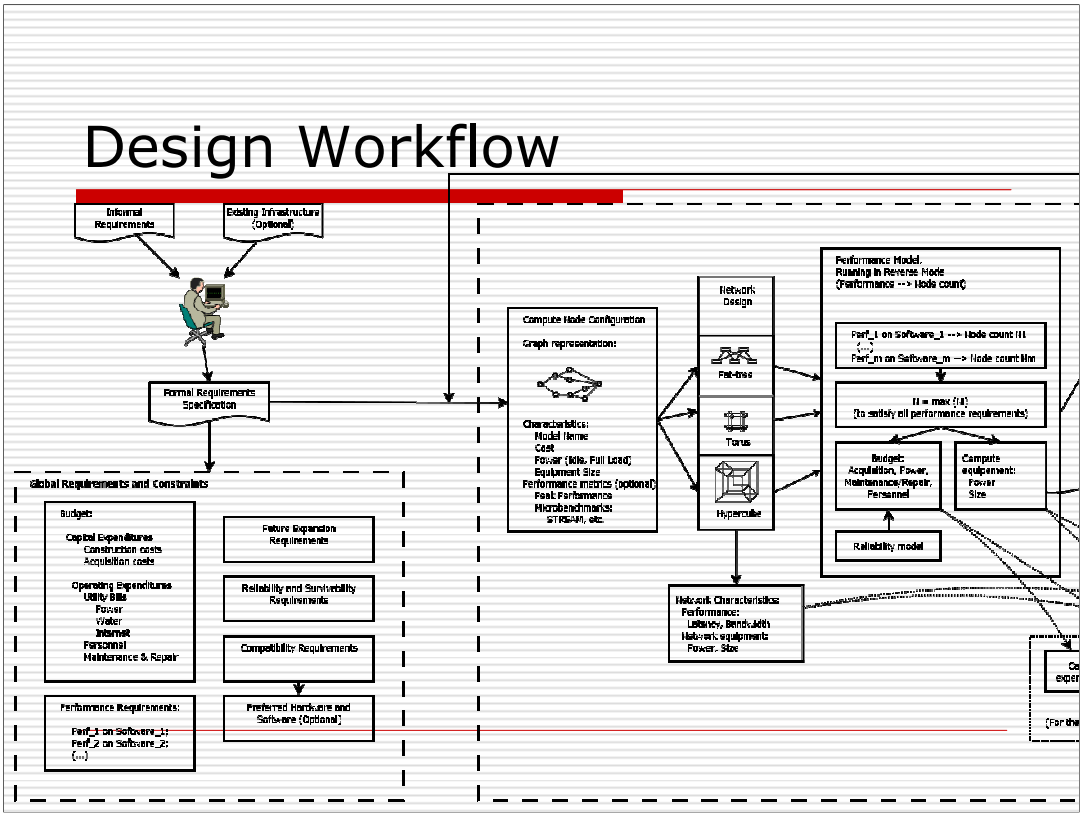
1. On the first stage, we query direct performance model, each time with the number of cores twice bigger than in the previous step (see gray dots on the curve), until performance becomes bigger than requested by the user, or the maximum reasonable number of cores is reached.
2. On the second stage, we search for the precise number of cores using the bisection method (see blue dots on the curve).

As a result, when a performance is specified, the corresponding number of cores can be found quite easily and in a few steps.

Design Workflow

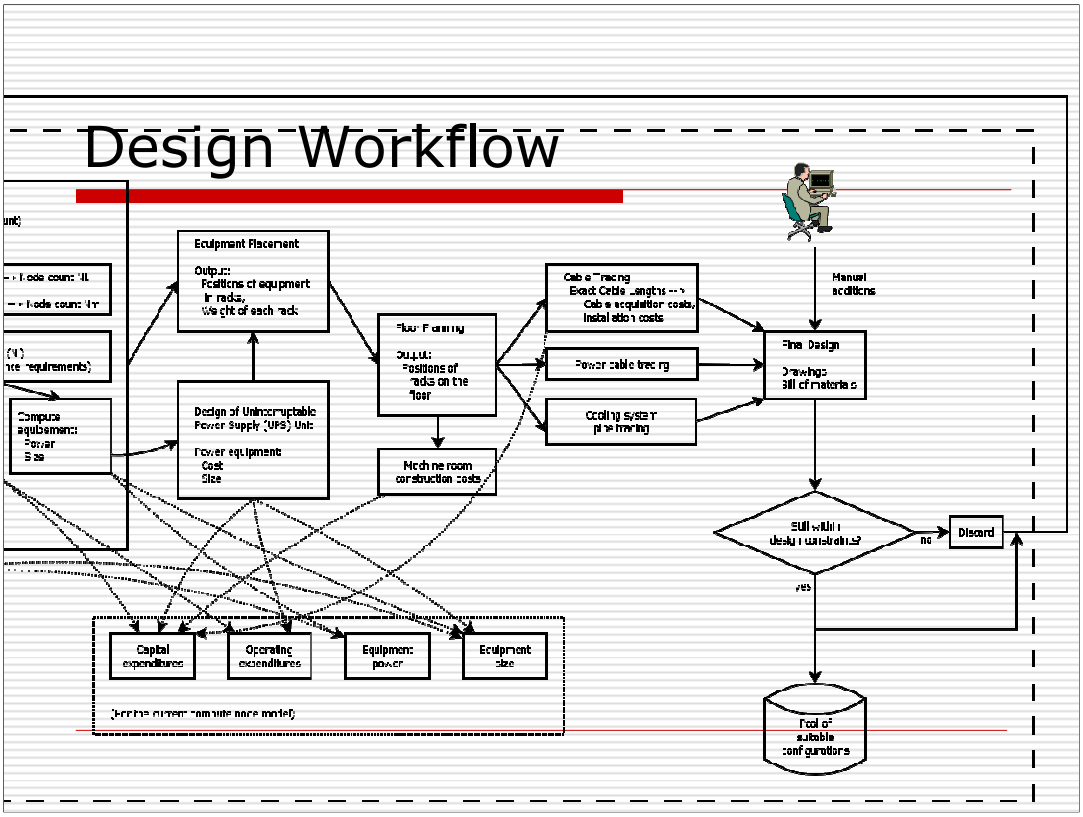


Let us now review the design workflow – the sequence of steps that must be taken to design a cluster supercomputer. As a diagram is rather big, we will review it in the two slides below.



1. Informal requirements specifications are turned into formal ones by a human engineer. This results in budgetary constraints (capital and operating expenditures) as well as performance constraints (i.e., performance with “Software_1” must be bigger than “Perf_1”, etc.)
2. Requirements for further expansion, as well as reliability requirements (useful for future exascale systems) can also be specified at this stage.
3. Compute nodes of a cluster are essentially servers, which can have multiple configurations (differing by the number of CPUs used, their model, the presence of hardware accelerators such as GPU, Intel MIC or FPGA, etc.) We iterate through all possible configurations of compute nodes. Configurations are represented with a graph, as outlined above, and by traversing the graph we obtain hundreds of configurations even for a single server model. (At this stage we can also exclude a particular configuration from further procedures by using heuristics)
4. For each configuration, we try different network designs as specified by the user. The choice of network heavily influences performance of the future system, and also has impact on supercomputer cost, power consumption and other vital characteristics.
5. The parameters of compute node, as well as the choice of network, are then fed into the inverse performance model, which gives us the number of compute nodes required to fulfill user-specified performance requirements. Armed with the number of nodes in a cluster, we can now update our budget (capital and operating expenditures) and check if we still fit within global constraints specified in the beginning. Power consumption of the machine and its size of equipment, measured in rack-mount units, is also updated at this stage.

(Switch to the next slide to proceed)

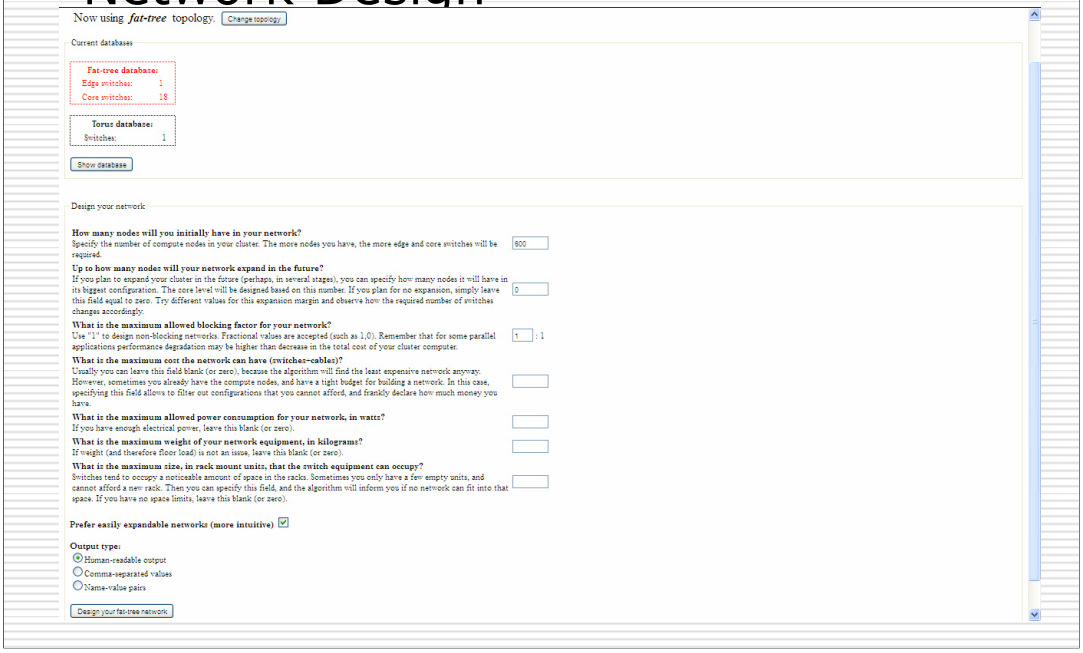


6. As we now know the power consumption of our machine, we can design a UPS system for it. Then, we update budget and other characteristics.
7. Then comes the stage of equipment placement into racks. We can fill racks as densely as possible, or only put a certain amount of equipment into racks so as not to exceed floor load. This is feasible, as weight of each equipment item is known. The placement problem generally translates to the famous “knapsack problem”, but a number of heuristics can be used to make this stage easier.
8. We know the number of racks, so we put racks on the floor, leaving enough space around them for maintenance purposes as specified by the vendor. This gives us the required floor space in square meters, which allows to estimate costs of renting this space or building a new machine room.
9. Network and power cables, as well as cooling pipes, can all be traced throughout the machine room by similar algorithms.
10. Final design contains necessary drawings and the bill of materials. It’s now time for a final check if we are still within global constraints. If yes, we add this design to the pool of viable designs, calculating for it the value of a certain objective function (“Total cost of ownership / Performance” appears to be a good choice)
11. We turn to the next configuration of the compute node, or, if all are exhausted, to the next model of a compute node. The design process then repeats again and again.

This may seem complex, but in fact, after all specific stages have been written down, it is just the matter of proper automation. And remember we can use modules to automate separate stages. A module for fat-tree and torus network design, as well as a direct and inverse performance model for ANSYS Fluent software, have already been created. They are accessible through a web interface and can also be used as helpful standalone tools.

Other modules (equipment placement into racks and rack placement on the floor, cable tracing, drawing creation and others) can be created in the future.

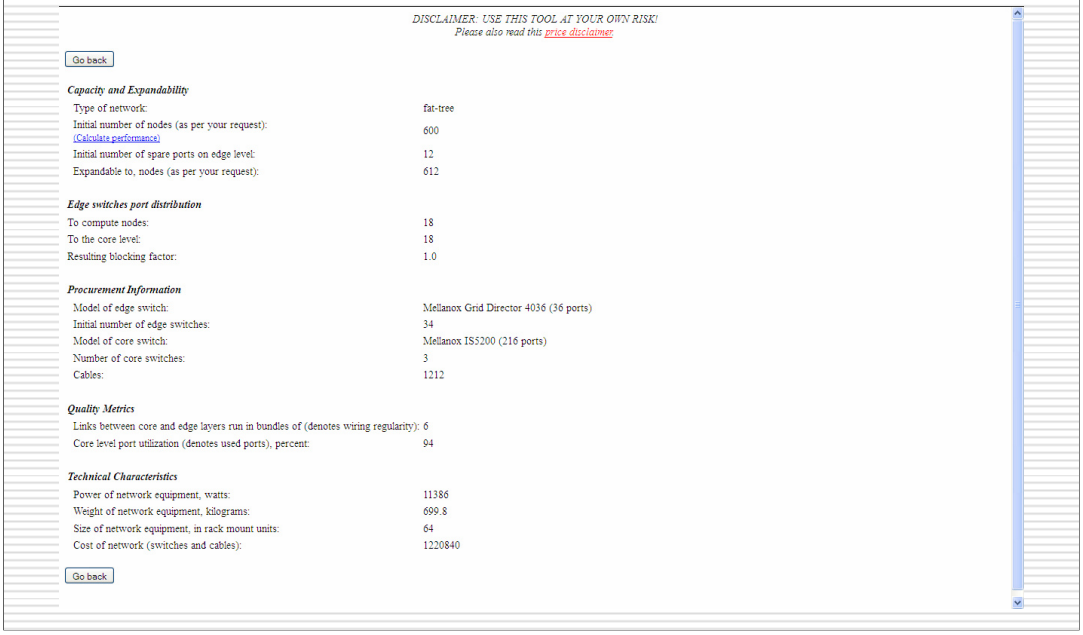
Web Service for Fat-Tree and Torus Network Design



The web service for fat-tree network design takes a number of parameters (such as the number of nodes that need to be interconnected) and constraints (such as the overall budget), and prints the results in several formats.

Internally, the web service tries all possible configurations of edge and core layers, using its internal database of switches, and returns the best configuration – the optimal one, according to a certain objective function. Currently, the built-in objective function is network cost.

Web Service for Fat-Tree and Torus Network Design



DISCLAIMER: USE THIS TOOL AT YOUR OWN RISK!
Please also read this price disclaimer

Go back

Capacity and Expandability

Type of network:	fat-tree
Initial number of nodes (as per your request): (Calculate performance)	600
Initial number of spare ports on edge level:	12
Expandable to, nodes (as per your request):	612

Edge switches port distribution

To compute nodes:	18
To the core level:	18
Resulting blocking factor:	1.0

Procurement Information

Model of edge switch:	Mellanox Grid Director 4036 (36 ports)
Initial number of edge switches:	34
Model of core switch:	Mellanox IS5200 (216 ports)
Number of core switches:	3
Cables:	1212

Quality Metrics

Links between core and edge layers run in bundles of (denotes wiring regularity):	6
Core level port utilization (denotes used ports), percent:	94

Technical Characteristics

Power of network equipment, watts:	11386
Weight of network equipment, kilograms:	699.8
Size of network equipment, in rack mount units:	64
Cost of network (switches and cables):	1220840

Go back

Here is the output in human-readable format. The optimal configuration is printed, including the number of switches and cables.

Most importantly, vital characteristics of the network – cost, size of equipment in rack-mount units, etc. – are all computed automatically and printed. This allows to call this web service from a CAD tool, and utilize its output in design procedures.

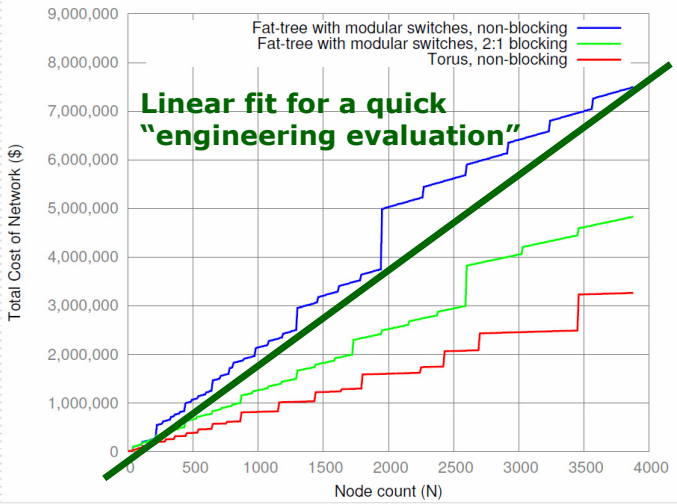
Web Service for Fat-Tree and Torus Network Design

```
nodes=600
nodes_future_max=600
max_network_blocking_factor=1.0
max_network_cost=0
max_network_power=0
max_network_weight=0
max_network_equipment_size=0
network_prefer_expandable=True
network_topology=fat-tree
network_edge_switch_count=34
network_edge_ports_to_nodes=18
network_edge_ports_to_core_level=18
network_links_run_in_bundles=6
network_core_switch_count=3
network_edge_uniform_distribution=False
network_link_count=1212
network_spare_ports=12
network_expandable_to=612
network_edge_switch_model=Mellanox Grid Director 4036 (36 ports)
network_core_switch_model=Mellanox IS5200 (216 ports)
network_cost=1220840
network_power=11386
network_weight=699.8
network_equipment_size=64
network_core_level_utilization=94
network_blocking_factor=1.0
```

← *Name-value pairs*

For automated invocations, the machine-readable output is the most convenient.

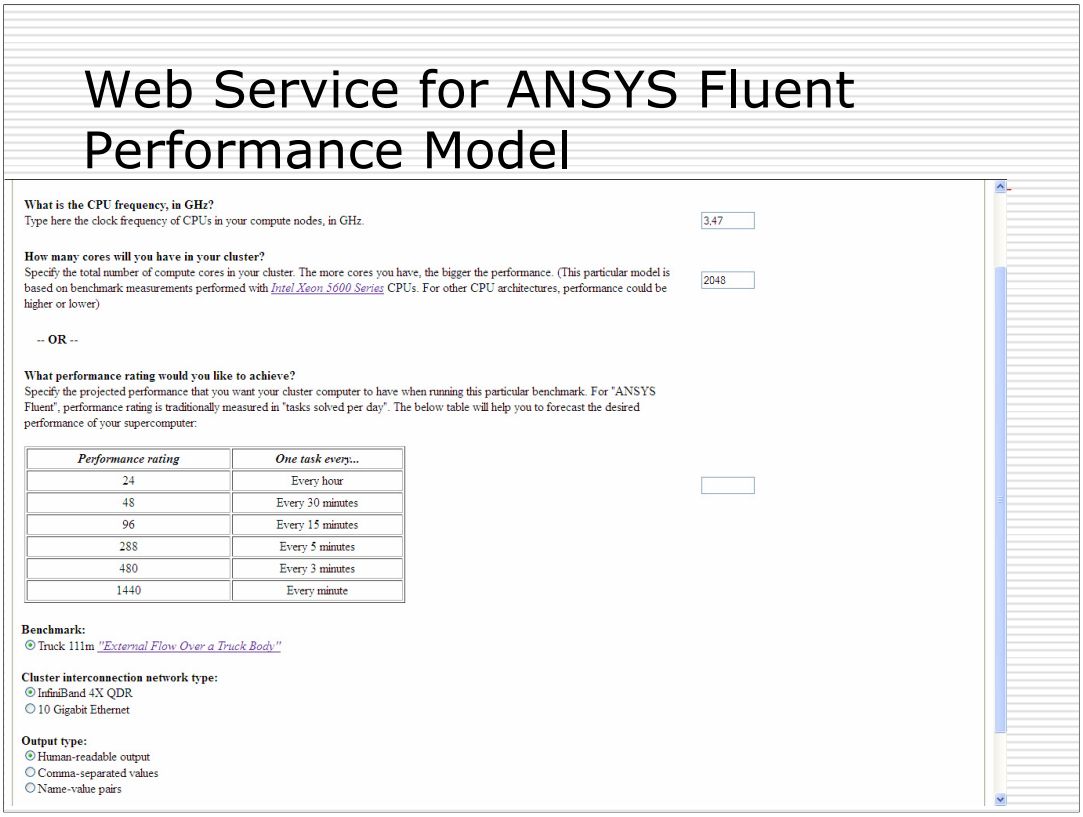
Web Service for Fat-Tree and Torus Network Design



We called this web service iteratively to calculate network cost for up to N=3,888 compute nodes. The results are on the graph.

Additionally, to a certain level of precision, a linear fit for network cost can be used as a quick “engineering evaluation”, as indicated by a green straight line.

Web Service for ANSYS Fluent Performance Model



What is the CPU frequency, in GHz?
Type here the clock frequency of CPUs in your compute nodes, in GHz. 3,47

How many cores will you have in your cluster?
Specify the total number of compute cores in your cluster. The more cores you have, the bigger the performance. (This particular model is based on benchmark measurements performed with *Intel Xeon 5600 Series* CPUs. For other CPU architectures, performance could be higher or lower) 2048

-- OR --

What performance rating would you like to achieve?
Specify the projected performance that you want your cluster computer to have when running this particular benchmark. For "ANSYS Fluent", performance rating is traditionally measured in "tasks solved per day". The below table will help you to forecast the desired performance of your supercomputer:

Performance rating	*One task every...*
24	Every hour
48	Every 30 minutes
96	Every 15 minutes
288	Every 5 minutes
480	Every 3 minutes
1440	Every minute

Benchmark:
(•) Truck 111m *"External Flow Over a Truck Body"*

Cluster interconnection network type:
(•) InfiniBand 4X QDR
( ) 10 Gigabit Ethernet

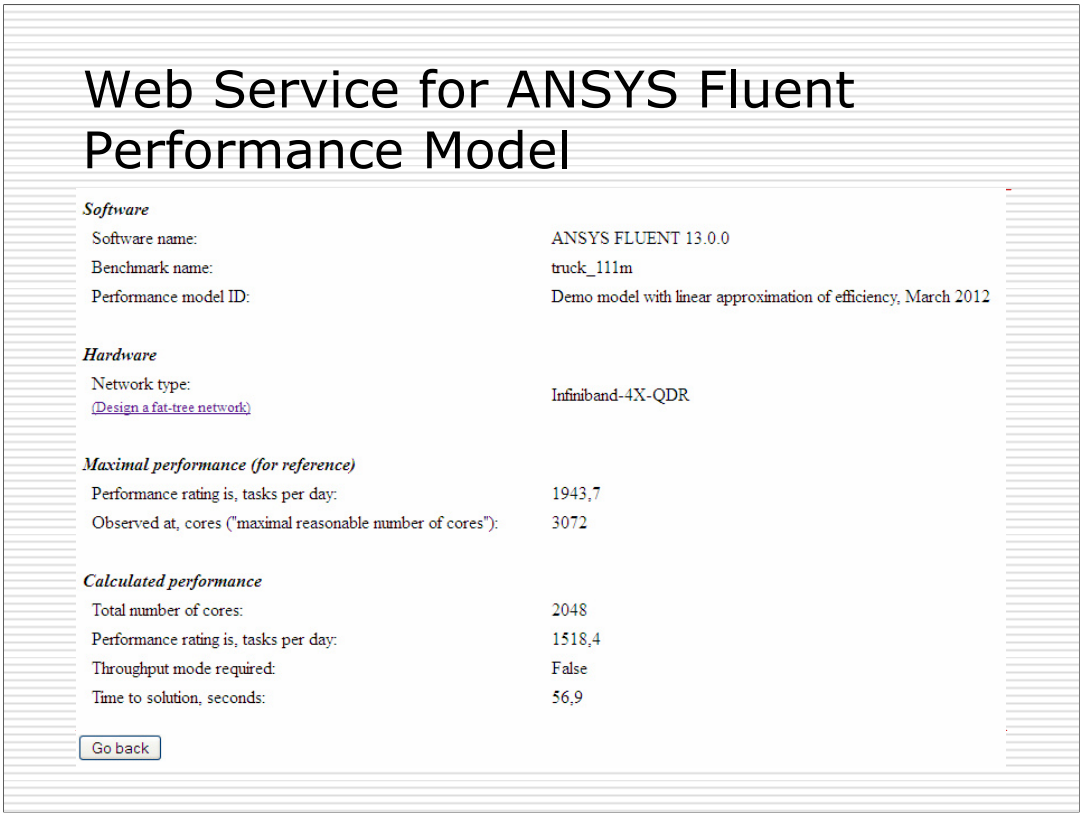
Output type:
(•) Human-readable output
( ) Comma-separated values
( ) Name-value pairs

The web service that implements the ANSYS Fluent performance model has a similar interface.

For direct performance modelling you need to specify the CPU frequency and the number of cores in the cluster, and the web service will return the performance evaluation. For inverse modelling, instead of the number of cores, supply the required performance rating, in tasks per day, and the number of cores will be determined automatically.

Another tuneable parameter is the network technology; and not surprisingly, InfiniBand gives better results than 10GigabitEthernet.

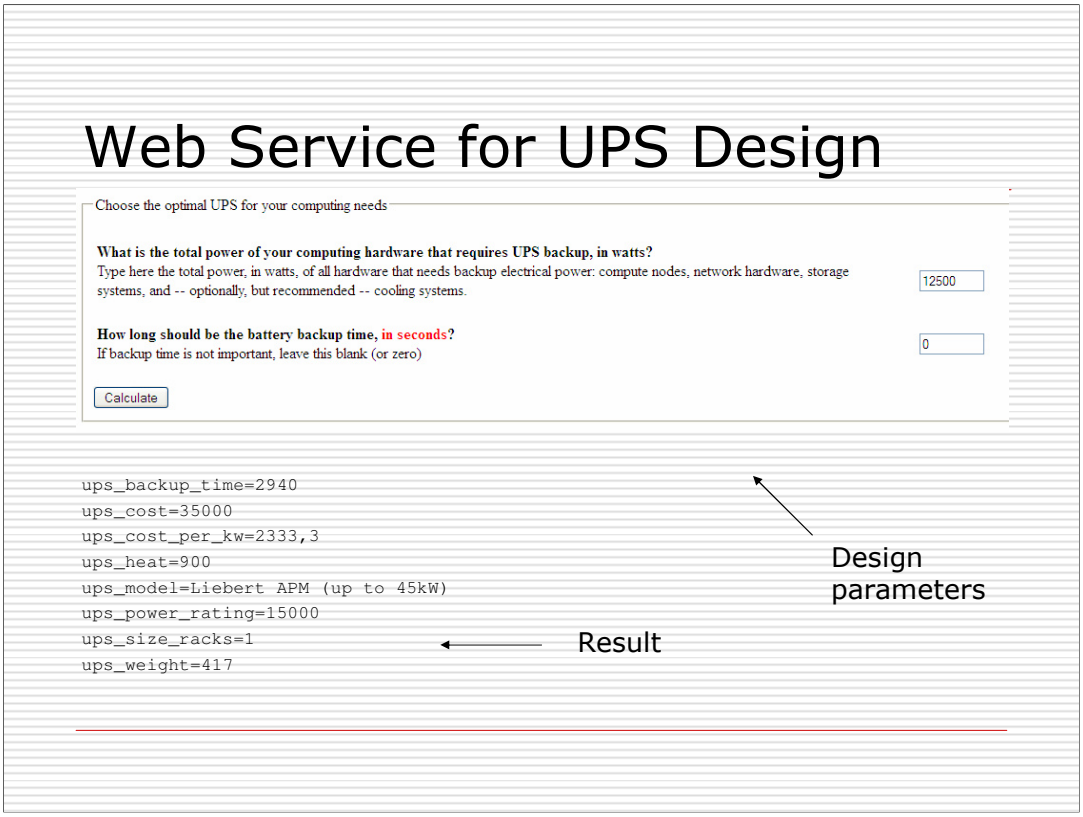
Currently, only one ANSYS Fluent benchmark was implemented, because sufficient amounts of benchmarking data were available for it.



Here is the sample output in human-readable format. The user supplied 2048 cores, and the resulting performance is 1518 tasks per day. Each 57 seconds a new task will be completed (“time to solution”).

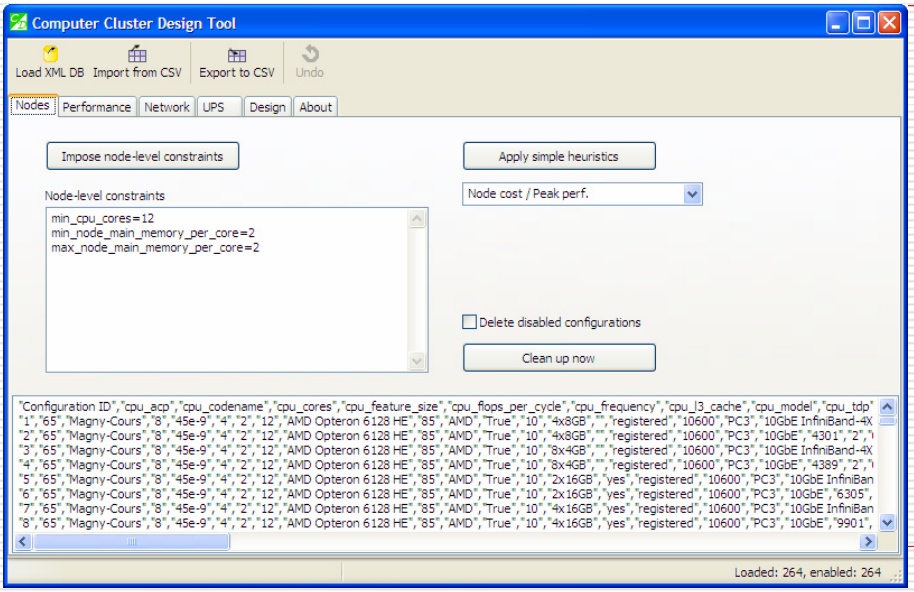
This result is for InfiniBand network. For reference purposes, maximal performance data is also outputted: the maximal rating is 1944 tasks per day, observed at 3072 cores. For comparison, the 10GigabitEthernet network technology only scales up to 384 cores.

Web Service for UPS Design



Along the similar lines works the UPS design web service.

CAD Tool for Computer Cluster Design



The CAD tool for computer cluster design was implemented. It utilizes web services described above.

The demo database of compute nodes contains a Hewlett-Packard's "HP BL465c G7" server, which has 264 different configurations. Note that we can significantly decrease the design space by applying heuristics.

The "Performance", "Network" and "UPS" tabs allow to load, correspondingly, performance, network and UPS design web services that will be used during the design process.

During the design phase, various technical and economic characteristics are evaluated (such as capital costs of computing hardware). Designs are then sorted according to the objective function (which is "Capital costs/Performance"), and best designs are printed.

For example, we asked the CAD system to design a cluster that would be able to achieve performance of 1600 tasks per day on ANSYS Fluent "truck_111m" benchmark. Out of 264 designs, the tool identified 4 designs of different cost. The best design had the following characteristics:

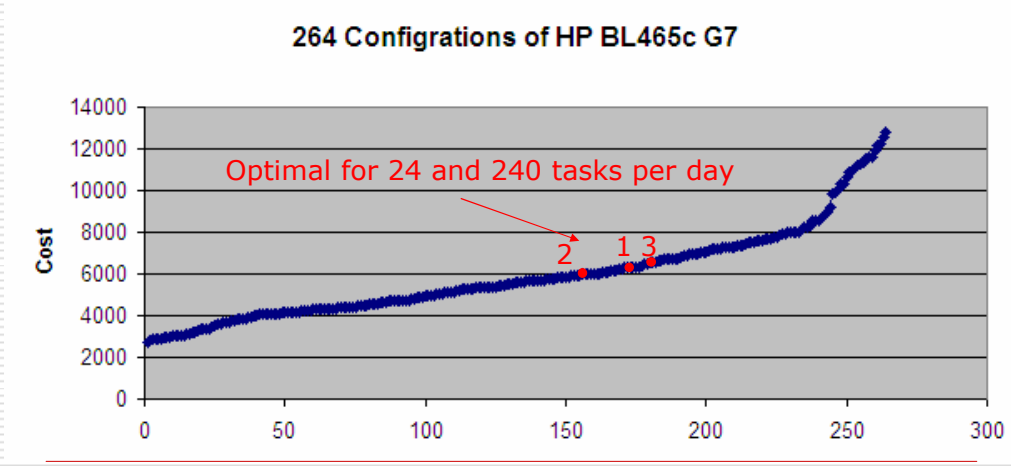
Compute node model: Hewlett-Packard BL465c G7

Cost: 1,374,300 US dollars

CPU model: AMD Opteron 6220

CPUs per compute node: 2

Analysis of Results

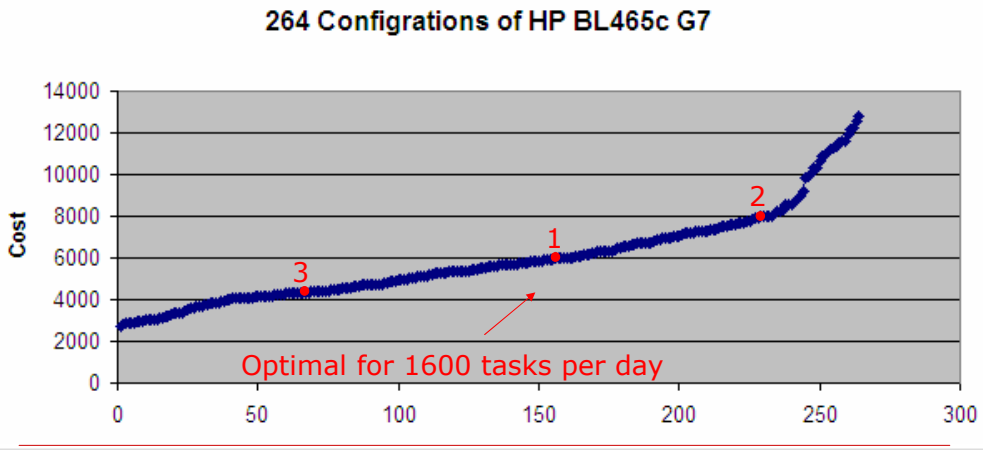


As we have 264 configurations of compute nodes based on HP BL465c G7 server, we plotted them on this graph. Every configuration has its own cost, which is represented on a vertical axis. As can be seen, configurations vastly differ in their cost, ranging from 3,000 to 13,000 US dollars.

When we run the design process with a goal of 24 and 240 ANSYS Fluent tasks per day, the following configurations (marked with 1, 2 and 3) are optimal. It is not the cheapest nor the most expensive configuration that is optimal. This strongly suggests that intuition alone is not enough to arrive to optimal designs.

We repeated this analysis with the goal of 1600 tasks per day, and the results are drastically different (see the next slide).

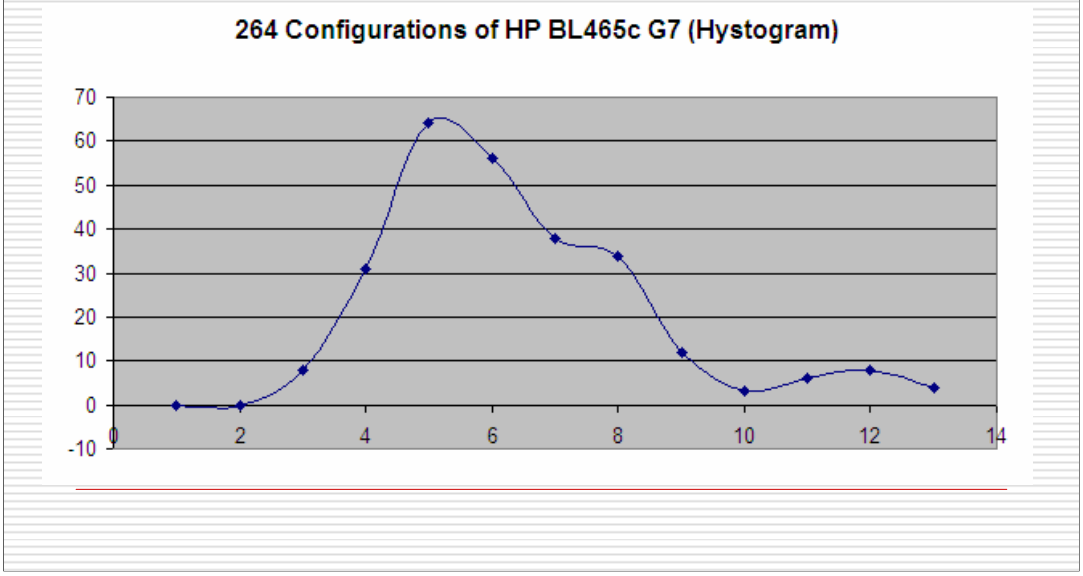
Analysis of Results



For 1600 tasks per day, the previous configurations were not optimal anymore, and new three leaders (marked with 1, 2 and 3) are now the best.

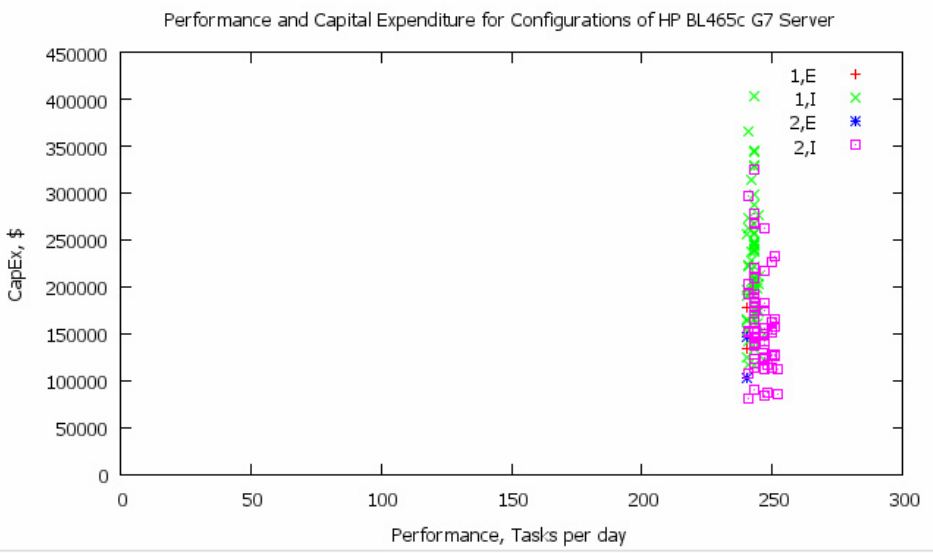
Interestingly, the best configurations of compute nodes all have very different costs. Therefore, one cannot infer whether the compute node configuration is good or not by only judging by its cost. This is yet another reason for design automation.

Analysis of Results



Here is another representation of 264 configurations of HP BL465c G7 server, this time using a histogram. A skew can be seen, which indicates that there exist more “costly” configurations than there are “inexpensive” ones; however, as was seen before, the optimal configurations can have any price which cannot be predicted beforehand.

Analysis of Results

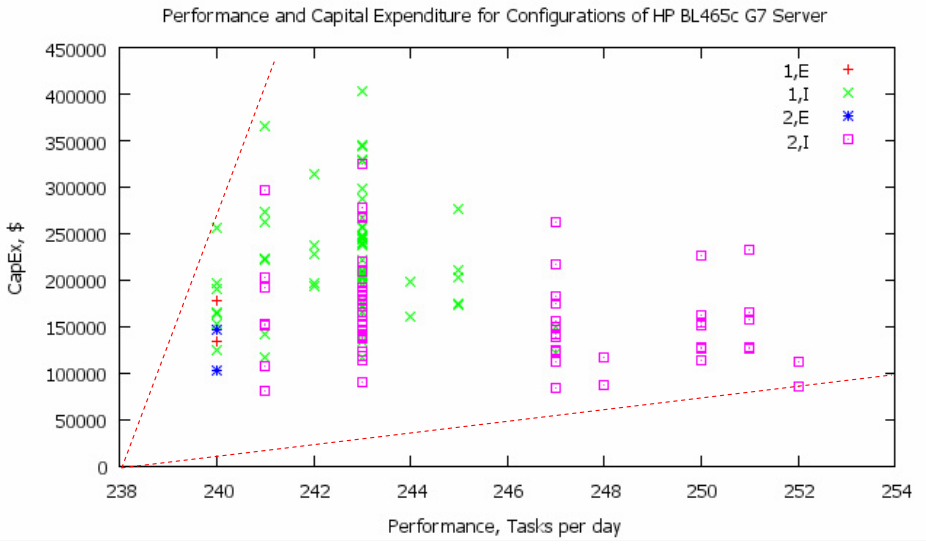


We also selected a pool of about 150 configurations that all yield the performance of 240 tasks per day. However, they differ in capital costs of hardware that must be procured.

Points on a graph are coloured according to the number of CPUs in a compute node (1 or 2) and the network type (“I” for InfiniBand and “E” for 10GigabitEthernet), see the legend.

The close-up of the region of interest is shown on the next slide.

Analysis of Results



As can be seen, for some configurations of compute nodes, a supercomputer can be built for less than 100,000 US dollars, but other configurations require up to 400,000 dollars investment in hardware.

Certainly, we need to find the best configurations. The performance is on a horizontal axis, and capital costs are on the vertical axis. Therefore, if the objective function is “Capital cost / Performance”, then the best configuration is easily determined, because the line passing through that point would have the lowest slope. Here, the best system has 2 CPUs and InfiniBand network. The worst system has one CPU and, somewhat surprisingly, also InfiniBand network.

Conclusions

- The framework to automatically design cluster supercomputers is now available
- The tools are free, open-source and extensible
- Your company's hardware can be easily added to the tools' databases
- Invaluable for any pre-sales & technical teams, as it:
  - designs computer clusters to customer's requirements
  - works fast and automatically
  - filters out inferior solutions, highlights the best one

More info and updates will be posted at http://ClusterDesign.org

Thank you for your attention!