Optimization Methods for Allocation of Large-Scale Computational Resources

Krzysztof Rządca

Where is Szczebrzeszyn? What is the fastest route to get there? Will it rain during the weekend? And how will the climate change in the next century? Is this signal a gravitational wave emitted during a collision of two back holes? Or just a microwave heating lunch in staff's cafeteria? We increasingly rely on computers to look for answers for more and more varied questions.

Hidden behind the neat web interface of a weather forecast are computational models painstakingly simulating the changing meteorological conditions. If it is 28°C and humid now, and 5km south-west from here there is a summer storm, what is the expected cloud layer in the next 5 minutes? What will be the wind? And the temperature? By solving mathematical formulas and repeating such calculations for subsequent time moments and for various locations, we end up with a reliable forecast for the next day or two.

These models require enormous computational power. The most involved models of climate, galaxies or systems of molecules take hours to solve when running on a *supercomputer* that has hundreds of thousands of processors and is roughly a million times more powerful than the laptop which I use to type this text.

The key to efficiency of a supercomputer lies in the scheduler: a software module that decides which jobs to launch on which processors (and which ones can be queued). Supercomputers constantly evolve in order to incorporate exciting new hardware such as accelerators based on high-end graphics card used to speed-up some calculations; or fast, but relatively small-capacity SSD disks. The scheduler policies must keep up with this new hardware.

While supercomputers run modern science, *data centers* run the modern internet, hosting services as diverse as Google maps, Facebook, WhatsApp, Flickr or DropBox. Data centers have similar architecture (thousands of computers connected by a fast network and managed by a common system) and face similar problems as supercomputers; additionally, they tend assign many jobs to each computer. As these jobs compete for shared resources, their performance varies considerably—which may result in slow or unresponsive services.

We plan to develop new algorithms for schedulers that replace heuristics and rules of thumb with optimization methods having mathematically-sound performance guarantees. We plan to address two main problems: first, managing the capacity of SSD disks in supercomputers to make individual jobs faster without introducing idle time in the schedule; and, second, allocating multiple jobs to a single computer in a data center in a way that they minimally influence each others' performance.

Our results should lead to more efficient use of existing resources, reduction of costs and, consequently, leading to their wider availability. In supercomputers, less idle time translates to faster results, shorter research cycles and, therefore, more meaningful science. In data centers, reduced performance degradation leads to a platform that is more robust and more stable, thus suitable for even more applications.