Research project: **Scalable Dispatching Policies**
Keywords: HPC, computing clusters, cloud computing, Markov chains, queueing theory
Supervisor: Prof. Esa Hyytiä (esa@hi.is)

Background: These days, most if not all popular services provided by the Internet-age giants such as Google, Amazon and Facebook rely on cloud computing. Consequently, the distributed system consists of a huge number of networked computing resources, and their efficient use is the cornerstore for a successful business operations.

To this end, researchers study appropriate models to understand pros and cons in different designs. One such model is a dispatching system, which consists of a dispatcther (or dispatchers) that distribute the computing jobs to different servers immediately upon arrival. The decision on server is made by the *dispatching policy*. The most efficient dispatching policies first query all servers, analyze their states, and then forward the job to the most suitable server. In large systems, with high volume of relatively short jobs (e.g., web requests) this is not possible. Hence, the dispatching decision must be made fast, preferrably without any queries.

Project description:

The most scalable policies are *static* policies, which simply analyze each job (e.g., its class, who submitted it, and size if available) and then forward it. For example, policy that chooses the server in random is static. The second "best" option is policies such as Round-robin, which store some information about the past decision in the dispatcher itself. This information is always readily available and thus does not slowdown the decision making much. The third option is to query only few servers, and then choose among them.

However, in real systems, the computing resources are often shared and multiple users run their virtual machines or microservices (e.g., Docker) in same physical hardware. This complicates the matters more as an end-user (customer) is not aware of the other users, internal priorities or scheduling policies.

The fundamental question is how to utilize the resources allocated for you most efficiently? Typical performance metrics are the mean response time and the probability that a job is not processed in time (quality of experience).

The tentative project plan is as follows:

1. Literature survey; get familiar with the past work and terminology
2. Analyze the round-robin when the inter-arrival time distribution ranges from almost constant to highly variable (bursty) cases. Both analytical and numerical approaches can be exploited (incl. simulations).
3. The so-called power-of-two policy chooses two servers in random and then applies JSQ. This has been shown to improve the performance drastically. However, better results can be expected if the subset of servers are chosen more intelligently. What is the optimal way, e.g., when the job sizes are known?
4. Redundant jobs is a concept where each job is sent to several servers. The copy completed first is used. Other copies are then deleted (if possible). The idea is that each job "finds" the right server this way. The downside is the increased load. This type of operation can be considered also.

The above list should not be taken literally. This is a genuine research project that will adapt to the person and take its own course during the work.

Prerequisites:

- Basic programming skills (Python, C and C++)
- Not afraid of statistics and probability theory
- Keen to understand how real computing systems can be modelled
- REI503M *Performance Analysis of Computer Systems* (highly recommended, lectured this fall)