Computational Cluster

Software Engineering 1

Krzysztof Kaczmarski
Michał Okulewicz

December 11, 2014

Abstract

Computational Cluster organizes and executes time consuming computational tasks. Especially for exact optimization algorithms with $o(2^n)$ time complexity.

The architecture of a system and a communication protocol design should focus on maximizing the effective usage of a resources (i.e. usage of computational power).

1 Glossary

Task manager - divides tasks for *computational nodes* to compute and chooses the best solution from all the *partial solutions*.

Computational node - computes the partial problems and sends *partial solutions* for them to the *task manager* through the *communications server*

Communications server - receives and dispatches *problem instances* and *partial solutions* to *task managers* and *computational nodes*.

Computational client - an application connecting to the *communications server* and sending to it the *problem instance* and waiting for the solution.

Problem instance - an identifier of a class of a problems (e.g. Travelling Salesman Problem) together with a specific instance (e.g. graph describing the cities location and routes between them).

Partial solution - the best solution for the *problem instance* found by a single computational node.

Task solver - module used by the *computational client* in order to find a *partial solution*

2 Task

The task is to design (till 30 January 2014) and develop (next semester) the architecture and communication protocol for the **Computational Cluster** and all its components (together with *Task solver* for Dynamic Vehicle Routing Problem).

3 Design requirements

Design documentation is to be prepared by 4 people teams. It must be sent to the teacher via mail in PDF format before the end of the semester. Please note that diagrams must be created with special treatment to be readable on a computer screen. The first page should contain a list of team members. The design documentation is a written document describing system activities, with **additional** clarifications presented in the form of the UML diagrams. All diagrams **must be** thoroughly commented and explained.

Required artefacts

1. Requirements Specification:
   (a) System actors' use case
      i. Person running a Communications server
      ii. Person running a Task manager
      iii. Person running a Computational node
      iv. Person running a Computational client
      v. Communications server communicating with task manager, computational node and computational client
      vi. Task manager connecting and communicating with communications server
      vii. Computational node connecting and communicating with communications server
      viii. Computational client connecting and communicating with communications server
2. Complete Design Documentation (Computational Cluster and Task Solver):
   (a) Components' runtime parameters / input data format specification
   (b) (High-level) class diagrams - describing structures, modules and architecture of the system
   (c) State diagrams - describing states of components of Computational Cluster and Task Solver for a given problem
   (d) Activity diagrams describing overall activities of the server and clients

(e) Communication protocol design with the usage of sequence diagrams and message description (suggested technology: XML Schema + examples of XML based messages)

(f) Additional relevant comments

(g) Special system states description (initialization, shut down, node failures) - in the case of such system special attention must be paid to the behavior after components' failures

3. Vocabulary (if needed)

4 Normal activity description

A task managers and computational nodes for a given class of problems register to the server. Server asks them about the name(s) of the class of problems that they can manage/solve.

Task manager

- sends names of the class problems to the server
- receives a problem instance to divide among computational nodes
- receives the number of computational nodes
- sends partial problems
- receives the partial solutions
- chooses and sends the final solution

Computational node

- sends names of the class problems to the server
- receives a partial problem to solve
- solves a partial problem
- sends partial solution to the server
- informs server that computation is ongoing

Computational client

- sends problem instance to the server
- receives the final solution

Communications server

- works in primary or backup mode
- sends problem instance to the task manager
- receives partial problems and sends them to the computational node
- informs other components of the system of existence of backup *Communication server(s)*
- manages states of computational nodes and assigned partial problems (send, computing, communication lost, ...)
- queues and (possibly) resends partial problems
- receives partial solutions and send them to the task manager
- receives final solution and send them to the computational client
- replicates its state (solutions and partial solutions, tasks to be computed or undergoing computations) to the backup *Communications server*

Task solver

- provides generic interface for *Computational node* and *Task manager* for solving, partitioning task and aggregating partial solutions
- loads (at runtime) libraries capable for processing different computational problems

5 Deployment scenarios

Figures 1 and 2 depict a simple and a complex version of deployment of system component over the hardware.

Figure 1 presents a simple configuration with 2 computational nodes, 1 task manager and a single client solving a SAT problem. Figure 2 presents a configuration with two clients, different kinds of computational nodes and many components are deployed on a single machine.

6 An example class of problems

An example class of problems could be a Dynamic Vehicle Routing Problem, a dynamic version of generalization of the Traveling Salesman Problem. The description of the particular type of DVRP, the Vehicle Routing Problem with Dynamic Requests could be found in [1].

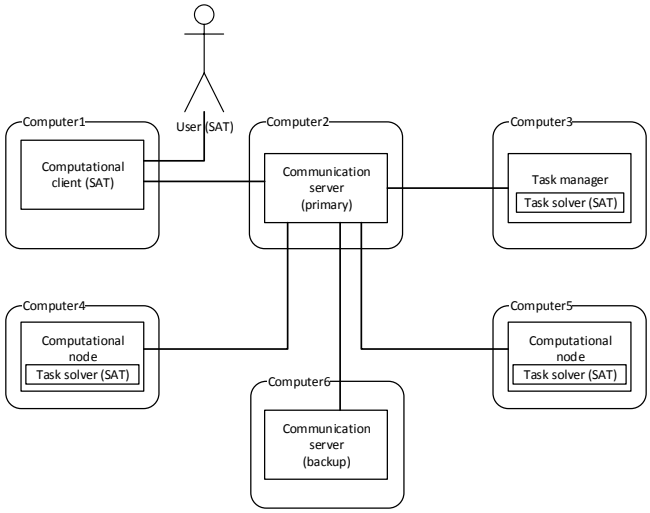

Figure 1: A simple deployment of components

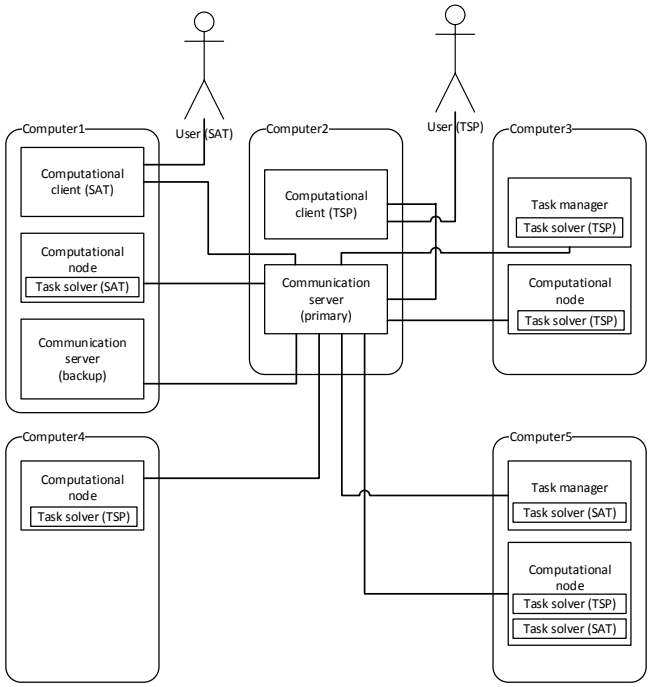

Figure 2: A complex deployment of components

References

[1] Mostepha R. Khouadjia, Briseida Sarasola, Enrique Alba, Laetitia Jourdan, El-Ghazali Talbi, A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests, Applied Soft Computing, Volume 12, Issue 4, April 2012, Pages 1426-1439, ISSN 1568-4946, http://dx.doi.org/10.1016/j.asoc.2011.10.023. (http://www.sciencedirect.com/science/article/pii/S1568494611004339)