P2PDisCo – Java Distributed Computing for Workstations Using Chedar Peer-to-Peer Middleware

Kotilainen N.[*], Vapa M.[+], Weber M., Töyrylä J. and Vuori J.
University of Jyväskylä, P.O.Box 35 (Agora), 40014 University of Jyväskylä
[niko.kotilainen, mikko.vapa, mweber, joni.toyryla, jarkko.vuori]@jyu.fi

Abstract

This paper introduces Peer-to-Peer Distributed Computing (P2PDisCo) software, which provides an interface for distributing the computation of Java programs to multiple workstations. P2PDisCo can be used to distribute any Java program that uses files for storing input and output parameters without significant code modifications to the Java program itself. P2PDisCo has been built over Chedar peer-to-peer middleware and is currently being used for speeding up the training of neural networks with evolutionary algorithm.

Keywords: peer-to-peer, distributed computing, Java, P2PDisCo, Chedar P2P middleware

1. Introduction

Peer-to-Peer (P2P) networks allow sharing of resources over the Internet. The resources can be for example, computing power, storage space, network bandwidth, printers etc. For sharing computing power, peer-to-peer networks are a natural choice, because many workstations are running idle most of the time.

In contrast to clusters, in P2P networks all the tasks and responsibilities for managing the network are shared between the peers. This means that there exists no single control entity responsible for providing the services. Also, because P2P networks do not require a dedicated hardware, distributing computation among workstations is usually a cost-effective solution.

Distributed computing on workstations is mostly known of SETI@home [1], which uses master-slave architecture for distributing the analysis of radio signals obtained from space to workstations. In this paper we present a peer-to-peer system, in which all the connected peers can work as master nodes initiating computations and also as slaves processing computations when idling.

The paper is structured as follows. Section 2 describes the related work in the area of P2P distributed computing. Section 3 introduces Chedar peer-to-peer middleware and section 4 the P2PDisCo peer-to-peer distributed computing software and its application programming interface (API). In section 5 we discuss the experiences gained from the use of P2PDisCo for distributing the training of neural network with evolutionary optimization algorithm and the planned future work. Section 6 concludes the paper.

2. Related Work

Nowadays there are many alternatives for distributed computing using Java programming language. One class of software is formed by programming language independent distributed computing tools that support Java. An example of such software is Globus Toolkit [2], in which Java Commodity Grid kit [3] provides an interface for accessing Globus services using Java programs. Globus contains mechanisms for code mobility which poses risks on security because the downloaded code needs to come from a trusted source or otherwise guaranteed to not be malicious. In our approach we have avoided the use of mobile code and the installation of the executed Java Archive file (jar) is done semiautomatically using copying scripts or by the user who donates computing power for the Chedar peer-to-peer network. Also Globus uses centralized indexes for resource discovery whereas in P2PDisCo the resource discovery is decentralized and provided by the Chedar peer-to-peer network.

Programming language dependent class of Java distributed computing can be divided in two: Java extensions and Java libraries. Java extensions such as JavaParty [4] provide special distribution mechanisms requiring changes to the Java compiler and/or Java Virtual Machine (JVM). This is a drawback considering the difficulty of executing the distributed code. Java libraries provide special class libraries for the distribution without

[*] The work of N. Kotilainen is supported by Innovations in Business, Communication and Technology (InBCT) project.
[+] The work of M. Vapa is supported by Graduate School in Electronics, Telecommunications and Automation (GETA).

need for modifications to the Java compiler or JVM. Therefore Java libraries are easier to deploy. An example of such library is JavaSymphony [5] as well as P2PDisCo presented in this paper. In JavaSymphony all the computing resources are centrally configured under JS-Shell whereas in P2PDisCo no central management exists.

There are also some implementations of Java distributed computing that use peer-to-peer network for locating the resources. In such design the resource index has been decentralized and peers cooperatively route resource queries among each other. An example of such system is GT-P2PRMI [6] which allows Remote Method Invocation (RMI) lookups to be performed through an extended version of RMIRegistry called P2PRMIRegistry. P2PRMIRegistry is used to form the overlay network, for binding and publishing the remote methods and for looking up the published remote methods.

3. Chedar P2P Middleware

Chedar (CHEap Distributed ARchitecture) is peer-to-peer middleware designed for peer-to-peer applications. Chedar constructs a pure peer-to-peer network using topology management algorithms and provides functionalities for locating resources in the network. The original goal of Chedar was to locate unused resources in a computer network that could be used for a given purpose; one could thus locate idle computers with a given characteristics in order to run computationally intensive calculations. It has then been extended to handle any type of resource: data (files), software (e.g. operating systems or specific applications) and hardware (e.g. computers, printers and displays). Implementation of Chedar is based on Java programming language, thus the software is platform independent and provides easy adaptation to different hardware.

Chedar nodes are identified with a pseudo-unique identifier called Chedar ID. Each node maintains a database of locally available resources shared by the owner of the device. These resources can include for example files and databases, software running on the device that can be accessed or used by remote users, and hardware characteristics of the device. Also, remote resources discovered on the network can be added to the database combined with information about their owner identified by Chedar ID and meta-information about themselves. Meta-information can contain e.g. type and path for the files, name and version for applications or any useful description for the hardware depending on the application, which is using the information. The resource database is stored as an XML document using a specific DTD. This organization of data allows making rich and complex queries to the database in the form of XPath expressions.

Chedar node keeps a list of neighbors it is connected to through TCP sockets. TCP provides reliable data delivery between the end points and thus also the disappearance of a neighbor can be detected. The neighbor list is updated based on heuristics such as the number of relayed query replies and the actual query replies provided by the neighbor to form an efficient topology for resource discovery. Currently, we use as a query mechanism breadth-first search algorithm (BFS), where query is forwarded to each neighbor except the one from which the query was received. Also, if the query has already been received it is not forwarded further. The number of hops that a query can take is limited in BFS with a time-to-live value. BFS is suitable for small-sized networks and guarantees to locate all resources from the network within the time-to-live horizon, but if the network grows larger the time-to-live value has to be decreased. In our experiments the network size of 200 workstations with 100 Mb/s Ethernet connections the query traffic has not yet posed a significant problem and therefore a more efficient version of the query algorithm has not been implemented.

Each query contains a Message-ID and a query XPath description. Whenever a query enters a Chedar node, the node checks its resource database whether it contains a resource matching the XPath expression. If resource is found, a reply message is sent back using the route, which the query came from. To properly relay the reply message back to the query originator, the message needs to contain the same Message-ID as the query did. After the query originator receives replies, it notifies P2PDisCo application, which can react to collected replies. Because the replies also contain the address of replying node, the intermediate nodes along the path as well as the querier learns new nodes in the network without being directly connected to them. This information can be later used for topology updates.

For communicating between two peers, Chedar provides a point-to-point communication protocol allowing basic message passing primitives to be executed by P2P applications. The protocol uses the same path as reply message to deliver messages between peers.

4. P2PDisCo

Peer-to-Peer Distributed Computing (P2PDisCo) was developed for distributing computationally intensive evolutionary optimization method to university workstations. The workstations are basically staying idle most of the time and therefore utilizing their processing power only at times when they are not in use does not interfere normal use of the computers.

P2PDisCo provides Distributed-interface definition with methods for starting and stopping the distributed application and checking whether the application is currently running. This interface needs to be implemented by the distributed application and is invoked by P2PDisCo.

The application is also required to read its parameters from a file and write its output to a file provided by P2PDisCo. The idea is that P2PDisCo pretends to the application that the application is reading all input data from files, but instead the file is delivered from Chedar and thus the application does not see a difference whether it is running remotely or locally. Also this ensures that the computing node does not need to store any data on its hard drive because Chedar delivers the data produced by application through TCP connections to the master node. After receiving data, master writes the received data to files on hard drive. The architecture of P2PDisCo and Chedar is shown in Figure 1.

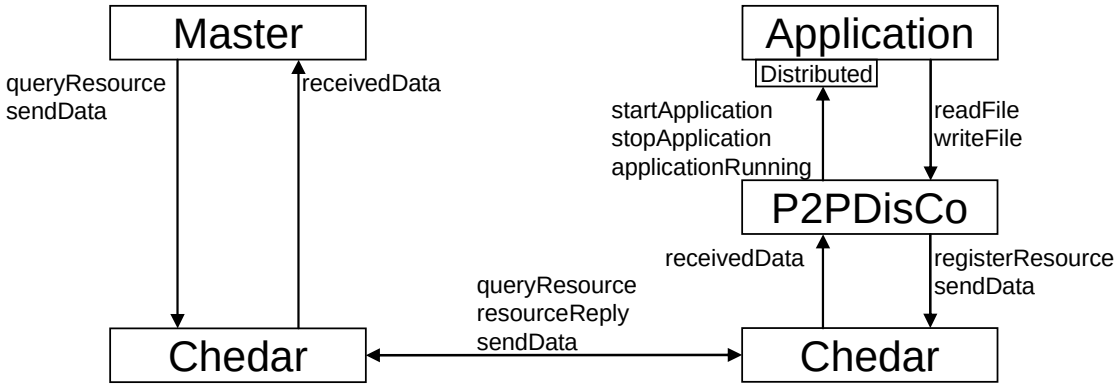


Figure 1. Architecture of P2PDisCo and Chedar.

The process of distributing the computation and collecting results using P2PDisCo is shown in Figure 2. When a peer (denoted as Master) joins the network, it needs to locate other peers to connect. This is currently handled by using a predetermined list of IP addresses and ports. If the peer has already been connected earlier to Chedar network it uses history data for connecting to already learned peers' addresses. Then master starts a query looking for idle computing resource and those peers that are ready for computing answer. Master selects which of the located nodes it uses for computing and distributes tasks to these nodes, which start the computing. During the computation, results are sent when memory buffer is full (currently at 256 KB) to master node and therefore no data is written to computing nodes. Also, this ensures that if the computing node is reset the computation results are still saved to the point of last full memory buffer update.

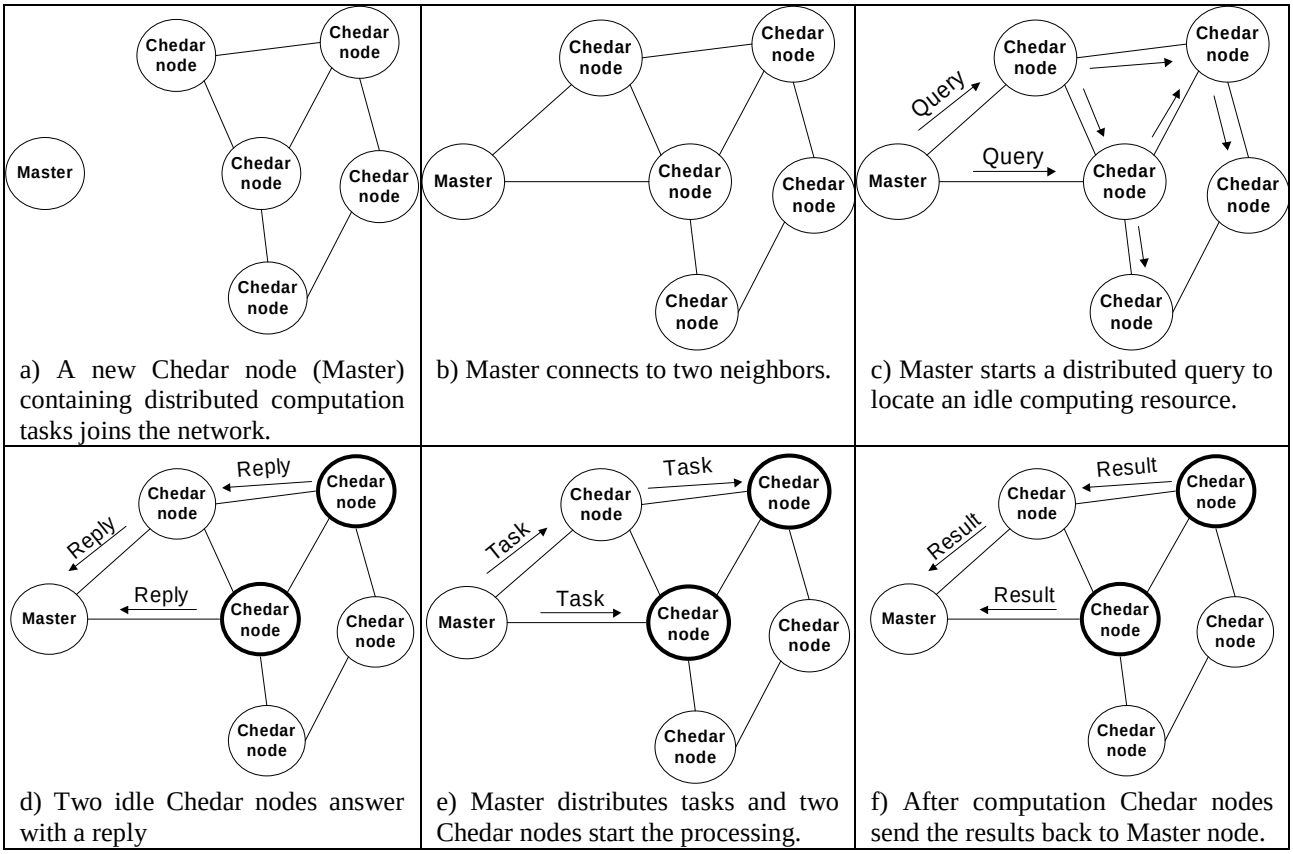
Because of security concerns the distributed application has been beforehand installed to the computers and it is not automatically delivered during the task distribution. In the task distribution only the execution parameters i.e. configuration files are transferred. Also, currently the IP addresses of master nodes are restricted such that only certain IP addresses are allowed to start computations.

5. Application Experiences And Future Work

P2PDisCo is at the moment used for speeding up the computations of NeuroSearch resource discovery algorithm [7] and it is deployed to more than 200 workstations of University of Jyväskylä in Agora building. NeuroSearch is a neural network algorithm, which is optimized using iterative evolutionary algorithm. It has been found that the evolutionary algorithm is a stable optimizer, but it requires much more computing power than for example back-propagation algorithms traditionally used for neural network training. The selection of evolutionary algorithm was needed because in the peer-to-peer resource discovery problem good input-output pairs are unknown and therefore no proper data for supervised training is available.

Based on half a year's usage of P2PDisCo it seems that the only major problem with P2PDisCo is the updating of the distributed application. Now it requires that Windows computers are updated with a script and the service running in Windows needs to be stopped and started again. At the moment there is no good solution how to avoid this. Fortunately, the computers are under central administration and the updating can be done from one computer.

As a future work, we are planning to add automatic resuming mechanism for computation, if computing node leaves the network. Now the computing is only restarted, but by checkpointing the state of current execution the computation could be resumed in another computing node from the point when connection was lost. Perhaps, in the future P2PDisCo also allows master to be disconnected for a while and collecting results afterwards from the computing nodes. Also, in the future we are extending the API of P2PDisCo to allow direct communication between computing nodes. This makes it possible to parallelize the evolutionary algorithm for multiple computers with other architectures than master-slave, such as the panmictic model [8].



a) A new Chedar node (Master) containing distributed computation tasks joins the network.

b) Master connects to two neighbors.

c) Master starts a distributed query to locate an idle computing resource.

d) Two idle Chedar nodes answer with a reply

e) Master distributes tasks and two Chedar nodes start the processing.

f) After computation Chedar nodes send the results back to Master node.

Figure 2. The process of distributing computation and gathering the results using P2PDisCo.

6. Conclusion

P2PDisCo provides decentralized architecture for distributed computing of Java programs. P2PDisCo is built on top of Chedar P2P middleware and requires only minor modifications to turn an existing Java application to a distributed one. Based on the experience of training NeuroSearch neural network algorithm, the system seems to perform well in a network of few hundred workstations. As future work resuming mechanism and extension of API to support direct communication between computing nodes is planned.

7. References

[1] SETI@home - The Search for Extraterrestrial Intelligence, http://setiathome.ssl.berkeley.edu/

[2] Foster I. and Kesselman C., "Globus: A Metacomputing Infrastructure Toolkit", *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2), 1997, pp. 115-128.

[3] Von Laszewski G., Foster I., Gawor J., and Lane P., "A Java Commodity Grid Kit", *Concurrency and Computation: Practice and Experience*, 13(8-9), 2001, pp. 643-662.

[4] Philippsen M. and Zenger M., "JavaParty: Transparent Remote Objects in Java", *Concurrency and Computation: Practice and Experience*, 9(11), 1997, pp. 1225-1242.

[5] Fahringer T., "JavaSymphony: A System for Development of Locality-Oriented Distributed and Parallel Java Applications", *IEEE International Conference on Cluster Computing*, 2000.

[6] Chang T. and Ahamad M., "GT-P2PRMI: Improving Middleware Performance Using Peer-to-Peer Service Replication", *10th IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2004.

[7] Vapa M., Kotilainen N., Auvinen A., Kainulainen H., and Vuori J., "Resource Discovery in P2P Networks Using Evolutionary Neural Networks", *IEEE International Conference on Advances in Intelligent Systems – Theory and Applications*, 2004.

[8] Alba E. and Tomassini M., "Parallelism and Evolutionary Algorithms", *IEEE Transactions on Evolutionary Computation*, 6(5), 2002, pp. 443-462.