Integrated Distributed Computing Environment on the G-Language GAE v.2

Ryo Hattori1,2 Kazuharu Arakawa1,2
hanzo@g-language.org gaou@g-language.org
Hayataro Kouchi3 Masaru Tomita1,3
haya@g-language.org mt@sfc.keio.ac.jp

1 Institute for Advanced Biosciences, Keio University, Tsuruoka 997-0035, Japan
2 Bioinformatics Program, Graduate School of Media and Governance
3 Department of Environmental Information

Keywords: distributed computing, G-language GAE, client server system, database management system

1 Introduction

G-language Genome Analysis Environment (G-language GAE) is a generic bioinformatics workbench aimed for higher efficiency in the process of analyses [1]. The open source (GNU General Public License) software package is available at the project website, http://www.g-language.org. Currently version 2 of the software is begin developed, which will be publicly released by Q2, 2004. The goals of the G-language Project are:

(1) Construction of an integrated environment for the development of analysis software.
(2) Systematic accumulation of the existing analysis software, methodologies for analysis and their results.
(3) Construction of generic analysis packages that allow users to avoid redundancy in the process of analysis.

In this paper, we report the development and implementation of the distributed computing environment utilizing database management system and client-server architecture. In order to analyze the huge genome data effectively and speedily, major improvements had to be made in the data management and calculation core. One solution is the construction of a distributed computing environment. The distributed computing environment on the G-language GAE is comprised of two components: the relational database (RDB)-based backend codenamed Bluebird, and the client-server based grid engine codenamed Infinity.

2 Method and Results

2.1 Implementation of the Database Management System: Bluebird

Large scale caching algorithm with relational database (RDB) is implemented as Bluebird. Bluebird takes advantage of DBI/DBD Perl modules and Tie function that enable flexible access to the RDB from Perl scripts, by tying the pointers of G-language GAE data structures to RDB schema instead of the RAM.

The concept of Bluebird is to meet the demands for greater efficiency and speed in analyses. For instance, in the case of analyzing huge human genome data on an assembly line, data management and calculation efficiency would be a problem. However, with the implementation of this caching
algorithm, *Bluebird* enables the users to manage data easily using RDB without stress, and to remove redundancies in disk access and time loss. Shown in Figures 1 and 2 are the speed and memory usage benchmark test compared between G-language GAE v.1 and v.2.

**Figure 1: Computational speed (s).**

**Figure 2: Memory usage performance (MB).**

### 2.2 Implementation of the Client-Server Grid Engine: *Infinity*

Client-Server based grid engine in the G-language GAE enables distributed computing without stress, and remove time loss. Distributed computing is a technique that makes computer able to deal with large amount of computation by parallel processing multiple CPUs. The protein 3D structure predicting software *Folding@home* and *mpiBLAST* are examples of distributed computing. *Infinity* server works as calculation server for computing arguments sent from the client that distributes the orders. The system includes security and load management features.

Figures 3 and 4 show that the speed and CPU performance benchmark test and machine specification (here named *Kevin* and *samaguchi*) We compared execution time of running a BLASTP search of the whole gene set of *E. coli* on one server and multiple servers. The result shows that the more CPU the less computational time. (In this figure, computational time of *kevin’s 1cpu = 1*)

**Figure 3: Computational time(s) & CPU performance.**

**Figure 4: Machine specification.**

### References