Towards Real-Time, Many Task Applications on Large Distributed Systems

- focusing on the implementation of RT-BOINC

Sangho Yi (sangho.yi@inria.fr)

Content

- Motivation and Background
- RT-BOINC in a nutshell
 - Internal structures
 - Design & implementation
- Conclusions and future work





Motivation



Demands for computing large-scale real-time(RT) tasks increased in distributed computing environment
Chess, Game of Go
Real-time Forensic Analysis
Ultra HD-level Real-time Multimedia Processing

Lack of support for RT in existing Desktop Grids, and Volunteer Computing environment

About BOINC

 BOINC is tailored for maximizing task throughput, not minimizing latency on the order of seconds.

XtreemWeb and Condor have similar characteristics.

A BOINC project has

A BOINC server (web, storage, database, ...)

Multiple BOINC clients

Network connection between server - clients

BOINC Projects SETI CHOME

- Normally perform a few transactions in 1 sec with host clients.
 - <u>1~50 transactions in 1 sec</u> (ref. <u>http://</u> boincstats.com)
- Send large chunk of computation to the host clients.
 a couple of hours, or even days of computation
- Does not have RT guarantee
 - Because it is tailored for maximizing total amount of computation.

Significant Gaps here...

"I need a 10-second-car." - in the movie "Fast & Furious"



Vin diesel – the main actor in the movie



Significant Gaps here...

"We need a 10-second-completion." - in a "Chess game"



Chess player's mission: Get next move within 10 seconds

REAL-TIME

RT-BOINC in a Nutshell

- RT-BOINC features
 - Providing low WCET (worst-case execution time) for all components
 - No database operations at run-time
 - O(1) interfaces for data structures
 - Reduced complexity for server daemons
 Almost O(1)

Original BOINC Internal





Data management

MySQL Database vs. In-memory data structures



Example 1) select from where;

Retrieving RESULT from the O(1) data structure



Performance Evaluation

- 1) Micro and Macro Benchmarks
 - Based on dummy server load
- 2) Case Studies
 - Game of Go AI, (and Chess AI soon)

Macro-benchmarks (high load)



Performance Evaluation - #2

Case Studies

Game of Go - 9x9 board (currently working)
 FueGo - a monte-carlo-based Al
 GTP protocol (go text protocol)
 KGS Go Server - can play with Al and human

Chess (developing with Emmanuel Jeannot)
 Distributed depth-first-search-based AI
 UCI protocol (universal chess interface)

Summary

RT-BOINC provides...

- Faster response time and real-time performance than BOINC.
- 300~1,000 times lower WCET(worst-case execution time) for each server-side operation.
- less difference between the average and the worst-case performance.
- less difference between low and high load conditions.

Future work (The rest part)



Future work (The rest part)



Go Al on RT-BOINC



Experimental Setup (1)

- We used a little bit fast machine, but used only 2 cores for this experiements.
 - We'll extend the scale of experiments when we have greater # of volunteers.

Component	Description	Notes		
Processor	2.00 Ghz (Dual-Quad)	Intel Xeon E5504		
Main Memory	32GB	(1,000 Mhz)		
Secondary Storage	HDD	- sorry for lack of info :')		
Operating System	Ubuntu 9.10 (karmic)	Linux Kernel 2.6.31-19		

Experimental Setup (2)

RT-BOINC

Up to 50k active wu, result, host, users

 3.9GBs of memory usage on a 64bit machine
 1.9GBs of memory usage for O(1) data structures (49.5 % of total)

BOINC

Recent server-stable version (Jun. 2010)

Minor Things for Experiments

- Apache & MySQL
 - Max # of connections (default is 100~256)
- Need 2 identical (physical) servers
 - For BOINC vs. RT-BOINC testing

Preliminary Results (Go Al)

Only preliminary results are available now.

Two cases: 160, and 480 cores (of volunteers)



Screen Shot on KGS



Difference of worst-case performance between low and high load condition



Performance Evaluation - #1

- Purpose: to measure real-time performance of BOINC and RT-BOINC
- Criteria: the worst-case and the average execution time
- Method: micro and macro benchmarks
 - Micro-benchmark: for each primary operation related to server process
 - Macro-benchmark: for each server process (including feeder, scheduler, transitioner, work-generator, assimilator, validator, and file-deleter)

Experimental Environment

■ We used a little bit slow, common-off-the-shelf system. ;-)

For ease of reproduction of the results

Component	Description	Notes		
Processor	1.60GHz, 3MB L2 cache	Intel Core 2 Duo		
Main Memory	3GB (800 Mhz)	Dual-channel DDR3		
Secondary Storage	Solid State Drive	SLC Type		
Operating System	Ubuntu 9.10 (karmic)	Linux Kernel 2.6.31-19		
BOINC version	Server stable version	Nov. 11, 2009 (from SVN)		

Average execution time (in seconds)



Worst-case execution time (in seconds)



Performance improvement ratio (RT-BOINC / BOINC)



Performance gap between worst-case and average



Macro-benchmarks (low load)



Source code on the Web

http://sourceforge.net/projects/rt-boinc

	Weld	come, Sangho Yi (ANTIROOT) Log Out Account
Find Software Develop Create Project Blog Site Support About		Q enter keyword Search
SourceForge.net > Find Software > RT-BOINC EDIT REAL-TIME RT-BOINC Beta by antiroot Summary Files Support Develop	EDIT	Share 🛃 눝 🚼 🖸 More 🛭 🍤 Donate
RT-BOINC stands for a Real-Time BOINC. It was designed for managing highly-interactive, short-term, and massively-parallel real-time applications. We implemented RT-BOINC on top of the recent BOINC server source codes. Download Now! Image: Column and the server s	EDIT	Rate and Review You gave this project a thumbs up! Image: Construction of the second
Image: boinc gridcomputing real-time rt-boinc rtbonc volunteercomputing Features: • ‡ Real-time server-side work unit transaction • ‡ Constant-time In-memory data structures (without using MySQL DB at ru • ‡ Using shared-memory IPC for both daemon processes (written in C) and	EDIT	Add an optional review: 30~100 Times higher performance than <u>BOINC</u> . 300~1000 Times lower <u>WCET</u> (worst-case execution time) for the given maximum load. Fantastic! Isn't it? <u>lol</u>
Compatibility with the original BOINC		Update Review > or Delete Review

Size of Data Structures

RT-BOINC uses the <u>'shared memory segment' IPC</u> between server daemon processes to share the data structures.

For 10,000 entries of hosts, results, workunits, it consumes totally **1.09GB** in main memory.

- Memory overhead for O(1) data structures is 38.6% of the total usage.
- Using **1GB** memory is reasonable on the common-off-theshelf 64-bit hardware platforms.

Detailed information on the Web

<u>http://rt-boinc.sourceforge.net</u>



REAL-TIME RT-BOINC stands for a Real-Time BOINC

It was designed for managing highly-interactive, short-term, and massively-parallel real-time applications. We designed and implemented RT-BOINC on top of BOINC server source codes. Contact information: Sangho Yi and Derrick Kondo

For users

Download RT-BOINC files Project detail and discuss Get support Donate money

For developers

Join this project:

To join this project, please contact the project administrators of this project, as shown on the project summary page.

Get the source code:

Source code for this project may be available as <u>downloads</u> or through one of the SCM repositories used by the project, as <u>page</u>.

About RT-BOINC

Future work (Remaining issues)

- Providing 'dynamic shared-memory management' to reduce memory usage
- Studying trade-offs between execution time and memory usage
 Studying better data structure management for O(1) response
- Finding better task deployment policy to
 - Reduce server-side load and latency
 - Improve real-time performance

Thanks! / Questions?





Example 2) insert into values(...);

■ Inserting RESULT to the O(1) data structure

Ex) insert into result values ();	
Get an available result field's id from end of list Then, remove the 'id' from end of list	
Lookup pool for available results	
Insert result to this place	
Pesult table	
(a) Insertion in main memory	

Example 3) delete from where;

Deleting RESULT from the O(1) data structure

Ex) delete from result where id='1234';		
Insert '1234' to the end of the result lookup list		
Lookup pool for available results		
Invalidate 1234 th result		
Result table		
(b) Deletion In main memory		

Prototype Implementation

Additional information

- Compaction of BOINC's data format
- Modification of PHP codes
- Trade-offs between memory usage and WCET
 Statically adjustable with parameters
- Compatibility with BOINC

The rest parts are still compatible with BOINC.