

*Sequence analysis***ANDY: a general, fault-tolerant tool for database searching on computer clusters**Andrew Smith<sup>1,2,3,4</sup>, John-Marc Chandonia<sup>2</sup> and Steven E. Brenner<sup>1,2,\*</sup>

<sup>1</sup>Department of Plant and Microbial Biology, 461A Koshland Hall, University of California, Berkeley, CA 94720-3102, USA, <sup>2</sup>Berkeley Structural Genomics Center, Physical Biosciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA, <sup>3</sup>Department of Molecular Biophysics and Biochemistry and <sup>4</sup>Department of Computer Science, Yale University, New Haven, CT 06520, USA

Received on August 22, 2005; revised on November 29, 2005; accepted on December 20, 2005

Advance Access publication January 5, 2006

Associate Editor: Martin Bishop

**ABSTRACT**

**Summary:** ANDY (seArch coordination aND anaYsis) is a set of Perl programs and modules for distributing large biological database searches, and in general any sequence of commands, across the nodes of a Linux computer cluster. ANDY is compatible with several commonly used distributed resource management (DRM) systems, and it can be easily extended to new DRMs. A distinctive feature of ANDY is the choice of either dedicated or fair-use operation: ANDY is almost as efficient as single-purpose tools that require a dedicated cluster, but it runs on a general-purpose cluster along with any other jobs scheduled by a DRM. Other features include communication through named pipes for performance, flexible customizable routines for error-checking and summarizing results, and multiple fault-tolerance mechanisms.

**Availability:** ANDY is freely available and can be obtained from <http://compbio.berkeley.edu/proj/andy>

**Contact:** [brenner@compbio.berkeley.edu](mailto:brenner@compbio.berkeley.edu)

**Supplementary information:** Supplemental data, figures, and a more detailed overview of the software are found at <http://compbio.berkeley.edu/proj/andy>

**BACKGROUND**

Many organizations are acquiring computer clusters in order to run large-scale biological database searches and similar applications efficiently in parallel over multiple cluster nodes. Unfortunately, while most researchers are able to run smaller database searches themselves on a single machine, it is not trivial to run such jobs in parallel on a cluster in an efficient and fault-tolerant way. ANDY is a set of Perl programs and modules that allows users to easily parallelize such jobs, and in general any sequence of Linux/Unix commands, on a cluster. Similar tools have already been written that are specific to particular search applications; e.g. TurboBLAST (Bjornson *et al.*, 2002) is a modified version of BLAST (Altschul *et al.*, 1990) that runs in parallel on clusters. More general-purpose tools such as Disperse (Clifford and

Mackey, 2000) and WRAPID (Hokamp *et al.*, 2003) allow users to specify a database search command line and have it run in parallel on multiple nodes of a cluster, which must be dedicated to the specific task. In contrast, ANDY sits on top of any cluster's general distributed resource management (DRM) and can intersperse fairly and efficiently with unrelated jobs. ANDY also provides key additional features and enhancements: extensive error-checking and fault-tolerance, simple configuration and extensibility to new applications.

**INFRASTRUCTURE AND CONFIGURATION**

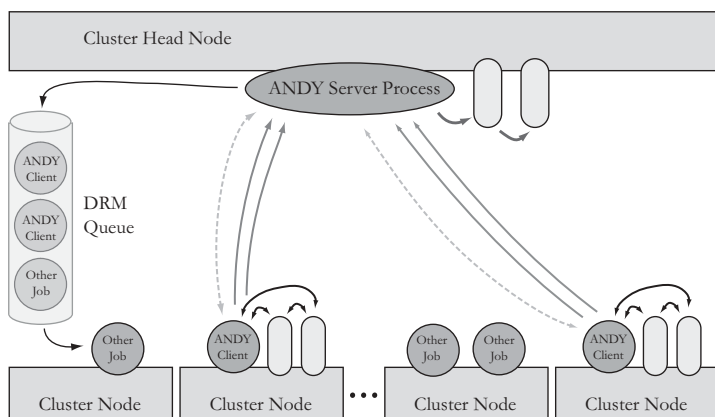
The ANDY infrastructure consists of a server process, started by the user on the cluster head node, and clients, which the server submits to the DRM to be run on the compute nodes (Fig. 1a). Each ANDY client process, upon starting, contacts the server to request configuration information for the run. Clients repeatedly request tasks from the server, interpolate a command template with values specific to the task, execute the task and send results and notification of errors back to the server. For example a single task might involve comparing a small number of sequences from one database against another. ANDY may be run in dedicated mode, in which a small number of client processes are submitted and once started on a compute node, they execute tasks until all tasks are completed, or in fair mode, in which each client exits after performing a modest number of tasks, and enough clients are initially submitted so all tasks will complete—in this latter case other non-ANDY jobs have a chance to intersperse with ANDY clients in the queue. Users configure ANDY through an XML configuration file that specifies a parameterized command template in common Unix shell syntax, along with the locations and types (e.g. FASTA file) of data sources that provide values for each parameter.

**FAULT-TOLERANCE**

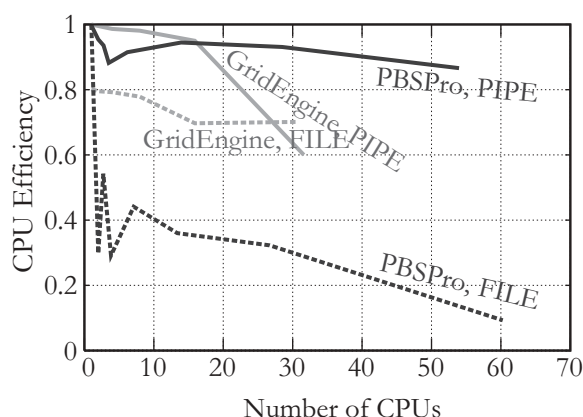
The ANDY server continuously monitors the DRM status of queued and running clients. Failed clients are restarted, and tasks that fail on one node are redistributed to other ANDY clients until a user defined

\*To whom correspondence should be addressed.

(a) Overview of ANDY infrastructure



(b) Scalability of ANDY on two clusters



**Fig. 1.** (a) Overview of ANDY infrastructure. ANDY components are shown in dark grey. The server process submits jobs, each containing one client process, through the DRM queue, where they may intersperse with other users' jobs (circles labeled 'other job'). Upon being started, clients communicate with the server (dashed arrows) to exchange configuration data and receive tasks. Clients start other programs (grey ovals) as specified in the command template to perform each task. Information is exchanged (black arrows) between these programs and the ANDY client through named pipes, memory buffers or files. One or more sets of results from each client may be sent back to the server (solid grey arrows), where they are saved to disk or optionally subjected to additional processing (grey ovals on head node). A color version of this figure is available in the online Supplementary information. (b) CPU efficiency for varying numbers of CPUs for a BLAST run with a fixed size search database, on two different clusters: one managed by the PBSPro DRM (dark grey) and one by the GridEngine DRM (light grey). The query database is a set of all protein targets from the Northeast Structural Genomics Consortium (Wunderlich *et al.*, 2004) in November 2004, about 10 000 sequences. The search database was a subset of sequences from the Pfam-A database of protein families (Bateman *et al.*, 2004), containing 531 384 sequences. CPU efficiency is the fraction of theoretically possible linear speedup achieved on multiple CPUs versus the lowest CPU time required to run a job on a single CPU on each cluster. The architectures of both clusters are typical: 32 dual-CPU nodes, where each node's two CPUs share common memory and local disk. It is probable that much of the performance disadvantage for files is caused by competition for the single disk on each node, as our PBSPro DRM schedules consecutive jobs on the same node, while the GridEngine DRM does not; this presumably accounts for the striking drop in efficiency when using FILES on the former cluster. Although both DRMs may be optimized to avoid competition for resources, use of PIPEs rather than FILES for inter-process communication gives a significant performance advantage in cases where such competition is unavoidable (i.e. a busy cluster). ANDY provides two implementations of named pipes: native (results shown) and memory-buffered. In the former case, Unix named pipes are simply substituted for files; the ANDY client manages their creation and cleanup, and interpolates the full path names into the command lines of tasks being executed. Although fast, this type of named pipe has several disadvantages: it cannot be randomly accessed and the contents can only be read once. In contrast, ANDY memory-buffered pipes are cross-platform, and allow data to be efficiently distributed to multiple tasks without being written to disk. As use of memory-buffered pipes incurs a performance penalty relative to named pipes, native pipes are preferred in cases where the additional flexibility is not required.

failure threshold is reached. The server does not exit until all tasks are completed. In many cases, task failure can be detected using Unix error codes; more generally, modules may also be written to detect application-specific errors. ANDY also monitors clients by listening for periodic signals from them. The server determines which clients should be resubmitted based on the job status history obtained using the client signals and the DRM, allowing reliable detection of job failure while minimizing unnecessary job resubmission.

## SUMMARY REPORTS

ANDY supports client-side pre-processing of results, such as extracting *E*-values from database search output, in order to limit the use of server disk space and network bandwidth and to parallelize the reporting. The server can save results it receives from clients directly to disk, or may optionally pipe the results into a user-specified command pipeline that executes on the head node throughout the run. This method of server-side processing is useful for creating a global summary of results (e.g. a summary of all search results sorted by statistical significance).

## PERFORMANCE

A key improvement of ANDY over similar tools is the support for input, output and inter-process communication through named pipes, in addition to files and unnamed Unix pipes. Pipes allow information to be passed in memory between consecutive steps in a pipeline of programs being run, rather than being written to disk. This can give a significant performance advantage, especially on typical clusters with multi-CPU nodes sharing common disk. In the performance tests that we have done with BLAST on two different clusters, one running PBSPro and the other GridEngine, named pipes provide a distinct performance advantage over files and allow ANDY to achieve nearly linear scaling in performance (90% CPU efficiency at maximum CPU usage) over nearly the full range of CPUs (Fig. 1b).

## FLEXIBILITY

Many clusters in the life sciences are managed and used through a DRM. Rather than integrating limited DRM functionality (as in similar tools such as Disperse and WRAPID), ANDY works seamlessly with third-party DRMs through modules. ANDY has been

tested on clusters running GridEngine, PBSPro, Ganglia/gexec and Condor. The DRM modules are the only code in ANDY specific to the DRM being used, and the tool is easily ported to new DRMs by writing a new module.

### ACKNOWLEDGEMENTS

This work is supported by grants from the NIH (R01-GM62621, 1-P50-GM62412, 1-K22-HG00056), the Searle Scholars Program (01-L-116), and by the US Department of Energy under contract DE-AC03-76SF00098. Hardware was provided through the IBM SUR program. Funding to pay the Open Access publication charges was provided by NIH K22 HG000056.

*Conflict of Interest:* none declared.

### REFERENCES

- Altschul,S.F. *et al.* (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
- Bateman,A. *et al.* (2004) The Pfam protein families database. *Nucleic Acids Res.*, **32**, D138–D141.
- Bjornson,R.D. *et al.* (2002) TurboBLAST: a parallel implementation of BLAST built on the TurboHub. *Proceedings of the IEEE International Workshop High Performance of Computational Biology*, April 15, Fort Lauderdale, FL, p. 1.
- Clifford,R. and Mackey,A.J. (2000) Disperse: a simple and efficient approach to parallel database searching. *Bioinformatics*, **16**, 564–565.
- Hokamp,K. *et al.* (2003) Wrapping up BLAST and other applications for use on Unix clusters. *Bioinformatics*, **19**, 441–442.
- Wunderlich,Z. *et al.* (2004) The protein target list of the Northeast Structural Genomics Consortium. *Proteins*, **56**, 181–187.