Reporting@Home: Delivering Dynamic Graphical Feedback to Participants and Researchers in Community Computing Projects

Stuart Ozer[+], David E. Kim[‡], David Baker[‡]

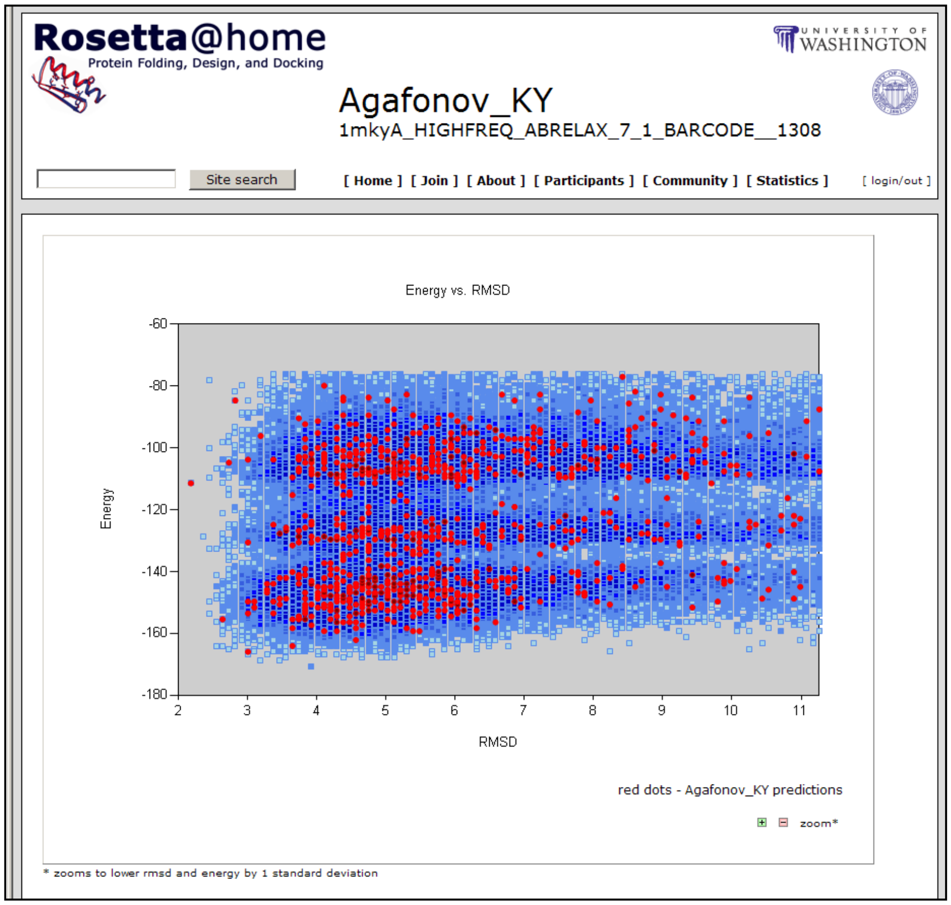
[+]Microsoft Research; [‡]Department of Biochemistry, University of Washington

ABSTRACT: A new generation of computationally intensive scientific research projects relies on volunteers from around the world contributing idle computer time to calculate mathematical models. Many of these projects utilize a common architecture to manage the scheduling and distribution of calculations and collection of results from participants. User engagement is critical to the success of these projects, and feedback to participants illustrating their role in the project's progress is known to increase interest and strengthen the community. This article describes how one project -- University of Washington's Rosetta@Home, which predicts and designs the folded conformations of proteins and protein complexes -- created a web-based, on-demand reporting system that graphically illustrates a user or team's contributions to the project. The reporting service is also useful to the project scientists in assessing the utility of alternative models and computational techniques. The system relies on a comprehensive database platform that includes tools for data integration, data management, querying and web-based reporting. The reporting components integrate seamlessly with the rest of the project's data and web infrastructure, and the report pages have proven to be popular among both participants and lab members.

1. Introduction

Community computing projects harness the CPU power of volunteers' idle personal computers or servers to perform large-scale distributed calculations for scientific research. Initially popularized by SETI@Home's signal processing project to detect signs of extraterrestrial communication [Anderson, 2002], current projects range from climate modeling, to searching for pulsars, to predicting folded conformations of proteins. BOINC (Berkeley Open Infrastructure for Network Computing) [Anderson2004] is a popular platform for implementing such projects – it has mechanisms for creating work units, distributing them to the community, executing them on client platforms, and collecting the results. Many popular projects use BOINC, including Rosetta@Home [Baker2006], [Rosetta] which predicts and designs the 3-dimensional structures of natural and synthetic proteins using minimum energy calculations. Although volunteers can observe the work that their computer is performing on behalf of these distributed experiments using the screensaver on their PC, until now there has been no mechanism available to visually present the overall results and the aggregate contributions of individual volunteers or teams.

Recently we enhanced the Rosetta@Home system to 'close the loop' with volunteers by providing on-request,



Agafonov_KY

1mkyA_HIGHFREQ_ABRELAX_7_1_BARCODE__1308

red dots - Agafonov_KY predictions

* zooms to lower rmsd and energy by 1 standard deviation

Figure 1. Scatterplot showing overall results (blue) and results specific to user (red).

internet-based reports that illustrate the contributions of individuals or teams, displayed in the context of overall results. The reporting functions are supported by a SQL Server relational database system [SQLServer] that tracks individual results and processes reporting queries on demand, and its associated Reporting Services that renders plots from query results – graphics that can be fully integrated with the project's public web site.

In Rosetta@Home, an individual computational run generates a folded protein conformation, along with two key metrics: *Energy* (Rosetta score of the computed protein structure which is analogous to free energy) and *RMSD* (Cα root mean square deviation from the experimentally determined structure). A typical experiment consists of generating a large number of conformations in an effort to find the lowest energy structure, which, ideally, should also have the lowest RMSD. A run's combination of energy and RMSD represents a single data point to be displayed by the reporting system. Figure 1 shows a sample report of many runs. The best folds are those at the lower left corner with low energy and low RMSD.

Since a single Rosetta experiment may have as many as one million result data points which must be presented graphically on-demand within a single chart, a combination of techniques and tools are applied in the reporting architecture to ensure that the system is both resource efficient and responsive to online ad-hoc requests over the internet. 1) Data points from each active experiment are fully refreshed in the relational database twice daily. 2) During the periodic database load, data points are pre-aggregated into 2-dimensional buckets appropriate for scatterplot rendering. Statistical metrics such as mean and variance are pre-computed per experiment to assist with automatic scaling of the plots. Additional aggregate information for each experiment is pre-computed per user, per machine, and per team. 3) The most popular reports overlay an individual user or team's results atop all other data points from the experiment so that a user can visualize their search space coverage and their proximity to the 'ideal' result. Since the rendering tool can display points in a 'layered' order, the result plot can be composed of two very efficient queries: a) one query that displays *all* results from the pre-aggregated 2-dimensional buckets and b) an overlaying query that displays a single user or team's contributions, projected onto the same buckets.

The feedback system has proven to be extremely popular with participants. We believe that similar techniques can be readily applied to other BOINC-based community computing projects and enhance participant interest in volunteering computer time for scientific projects.

The reporting system is also useful to investigators inside the Baker lab for quickly monitoring the progress of experiments and viewing results.

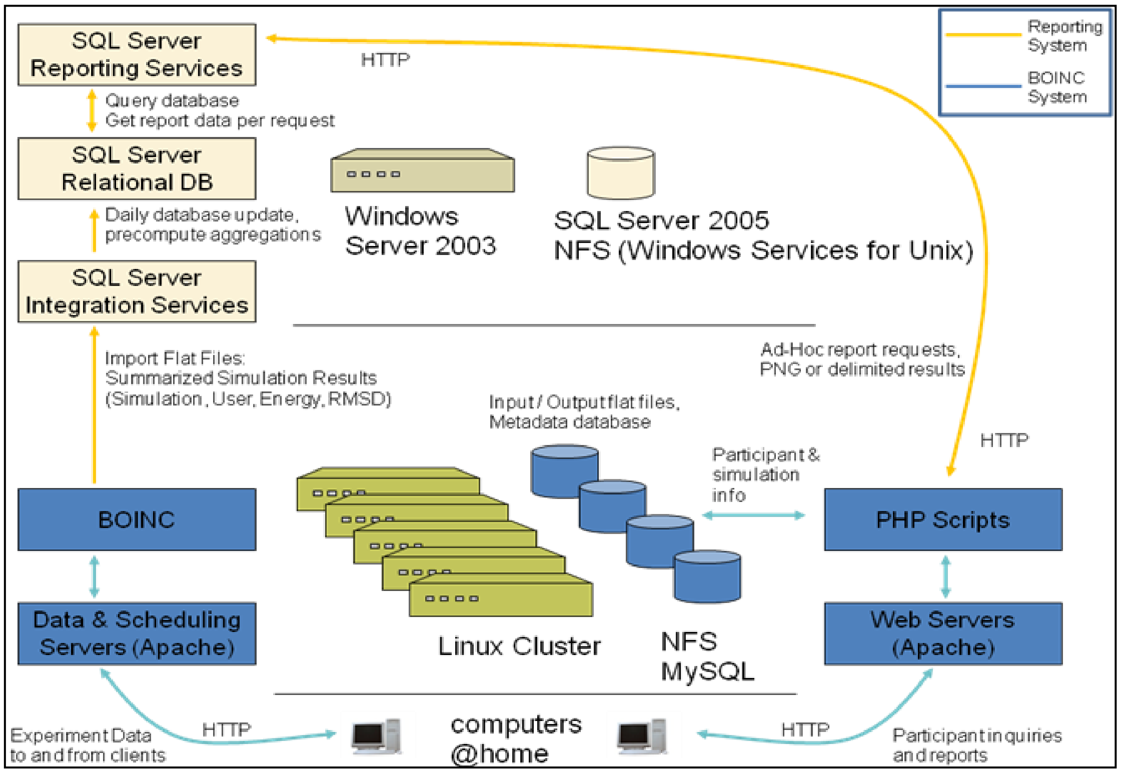

SQL Server Reporting Services
HTTP
Reporting System
BOINC System
Query database
Get report data per request
SQL Server Relational DB
Windows Server 2003
SQL Server 2005
NFS (Windows Services for Unix)
Daily database update, precompute aggregations
SQL Server Integration Services
Import Flat Files:
Summarized Simulation Results
(Simulation, User, Energy, RMSD)
Ad-Hoc report requests, PNG or delimited results
Input / Output flat files, Metadata database
Participant & simulation info
HTTP
BOINC
PHP Scripts
Data & Scheduling Servers (Apache)
Linux Cluster
NFS MySQL
Web Servers (Apache)
Experiment Data to and from clients
HTTP
computers @home
HTTP
Participant inquiries and reports

Figure 2. Overall system architecture and data flows.

2. Architecture

The BOINC system that orchestrates Rosetta@Home's distributed experiments is hosted on a Linux cluster utilizing flat files on a Network File System (NFS) store for retaining results from the community of participants. Of particular interest for the reporting system are the summary flat files, maintained for every experiment containing the Energy, RMSD, UserID, ComputerID, TeamID of every prediction. New data is appended to these files continuously by BOINC's assimilator daemon as participating computers return results to the lab.

The Reporting System and database is hosted on a Windows server running SQL Server 2005. It accesses the summary flat files from BOINC using a SAMBA share and imports the flat file data into the SQL database every 12 hours using SQL Server Integration Services – a visual workflow tool integrated into SQL Server that is designed to manage data extraction and transformation.

The Reporting System server also hosts SQL Server Reporting Services and Internet Integration Services (IIS) – the Windows-based web server. The IIS web server software accepts HTTP requests for reports and routes them to the Report Server software to render. The requests embed report parameters (such as UserID, Experiment ID [also known as SubBatchID], TeamID, etc.) as embedded parameters in the URL string. Report Server issues queries to the SQL Server database and renders results in a graphical format, returning tabular HTML or graphical PNG files to the requestor. These components run together on a dual-core, dual processor AMD server running at 1.8 GHz with 4GB of RAM and a set of high capacity mirrored disk drives.

The Rosetta@Home web site, hosting all public internet functionality for the project including reports, runs on an Apache web server with applications scripted in PHP. This is the only public-facing web server for the project. When reports are requested, the Apache server issues private HTTP requests to IIS running on the Reporting Server and handles the result set, whether text or graphics, under control of the PHP scripts. In the end, the integration between these two very different environments – *Windows / SQL / IIS* and *Linux / Apache* – is seamless. It is also secure, as the server and database supporting the reporting functionality is not directly accessible to public requests over the Internet.

3. Database

The database supporting the reporting system captures the essential details for each data point calculated by users, along with aggregates that support rapid reporting. The schema is shown in Figure 3. Rosetta experiments are known to the system as "SubBatches", identified by a *SubBatchID*. The table ***RosettaSubBatches***, keyed on *SubBatchID*, maintains identifying data about each experiment including the *SubBatchName* and *LastModifiedDt*

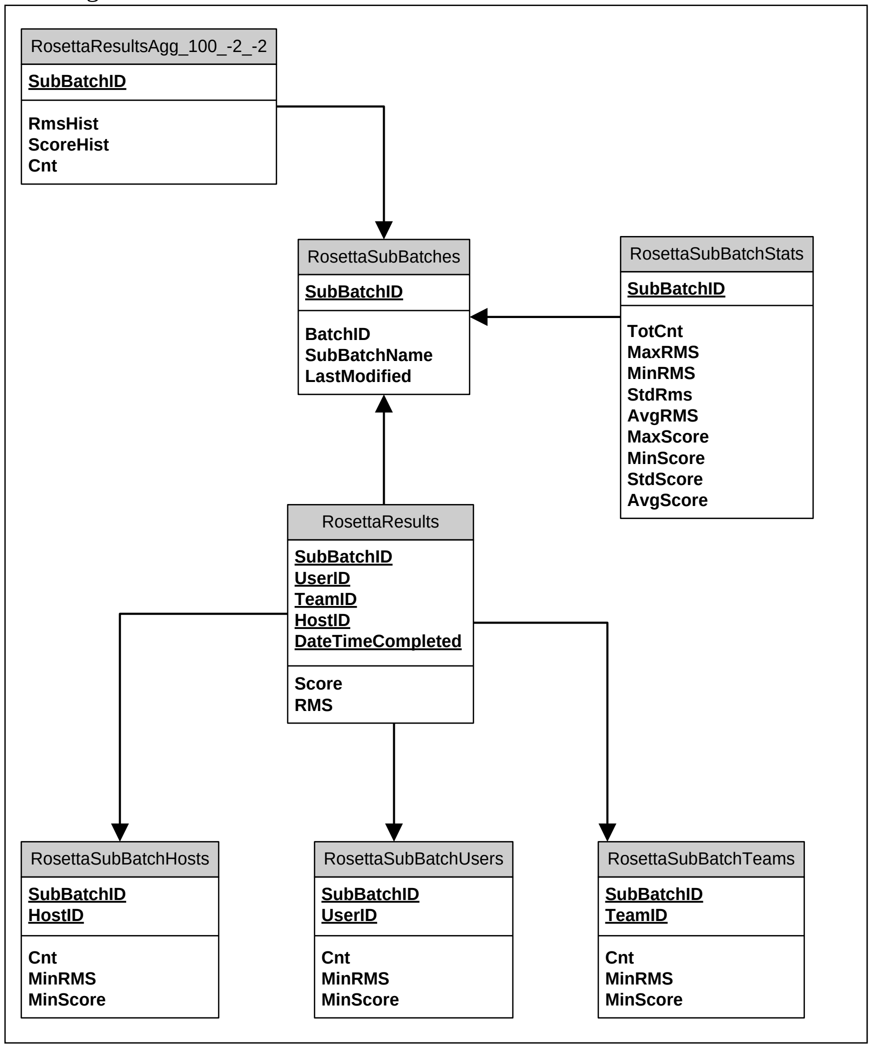


Figure 3. Reporting database schema

– used to determine when an experiment should be aged out of the reporting system. Thousands of users may be involved in computing results within a SubBatch. The core table storing these results is ***RosettaResults***, keyed on *SubBatchID*, *UserID*, *TeamID* and *HostID*. The columns *Score* and *RMS* retain the floating point values of the Energy and RMSD associated with that result, respectively. At the end of 2006 there were 3900 active Sub Batches and 115 million Results.

While in theory every report of interest could be generated using these two tables, we would be repeatedly scanning and aggregating results from SubBatches as well as computing scaling parameters for graphics (the coordinate locations of the graphs' corners). Because the reporting data is updated intermittently rather than continuously, it is useful to preaggregate much of this information.

To support the reports that rank users, teams and hosts, we maintain the summary tables ***RosettaSubBatchUsers***, ***RosettaSubBatchTeams***, and ***RosettaSubBatchHosts***. Each is keyed on *SubBatchID* and either *UserID*, *TeamID* or *HostID* respectively, and each contains data elements aggregating the result count for that grouping (*cnt*), as well as the minimum RMSD and minimum Energy (*MinRMS*, *MinScore*). A sample report generated using these aggregates is shown in Figure 4. Scatterplots displaying the range of experiment data points in RMSD / Energy space (as shown in Figure 1) offer another database optimization opportunity. Results are displayed in a rectangular 100 x 100 pixel grid, with each point shaded to reflect the count of results within that grid bucket. This reduces the required query size to one of delivering at most 10,000 counts (representing pixels), down from a possible 1 million or more individual data points. In addition we allow reports to specify the scale ('zoom' value) of the display by selecting the number of standard deviations above or below the mean to display as the maximum value, on both the RMSD and Energy axes. (The minimum value shown on any graph is always the minimum RMSD and minimum Energy data points for that SubBatch).

Thus all queries that generate the scaled data values for the 100 x 100 scatterplot always reference common statistical metrics for the sub-batch, including the Min, Max, Mean and Standard Deviation. To avoid repeatedly reading all data for a sub batch to compute these terms, we maintain the table ***RosettaSubBatchStats***, keyed on *SubBatchID*, to retain these precomputed values.

Finally, to accelerate the most common scatterplot queries, we maintain the nonzero total counts associated with each pixel in the 100 x 100 grid for the default chart -- that covers the range from the minimum values to 2 standard deviations beyond the mean – in a separate aggregate table named ***RosettaResultsAgg_100_-2_-2***. This table provides a quick rendering of the background data, upon which the detailed data for a single user or team can be superimposed (see Section 5).

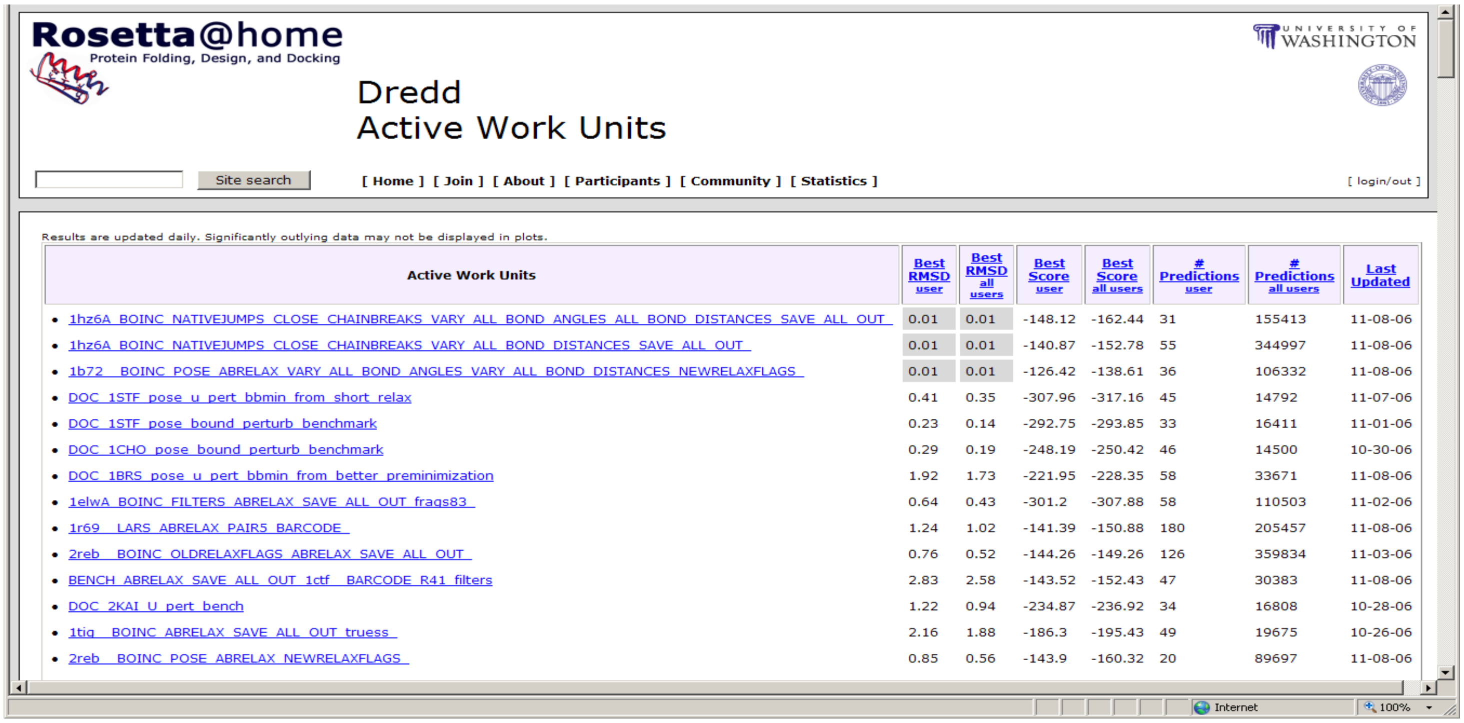
While all of these precomputed aggregate tables are not *necessary* to extract results for reports, their use significantly reduces CPU time and IO workload of the database server, allowing queries to execute faster and the system to scale more easily to support a large concurrent user base.

4. Populating Data

The daily refresh of data in the Reporting database is orchestrated by a SQL Server Integration Services (SSIS) package that refreshes information per-experiment. For each active experiment in the BOINC system,

- All data for that experiment (*SubBatchID*) in the ***RosettaResults*** table is deleted.
- All data associated with that *SubBatchID* in each of the aggregate tables is deleted.
- ***RosettaResults*** is loaded with refreshed data from the flat file of results for that *SubBatchID*, available in the NFS share from BOINC, using an SSIS Data Pump Task. This process uses the SQL Server Fast Load API.
- Queries are issued against the ***RosettaResults*** table, restricted to the current *SubBatchID*, to populate the aggregate tables using the `INSERT INTO … SELECT FROM` syntax.

The entire processing cycle above for a single experiment requires only a few seconds to complete, and the batches are processed serially to avoid locking conflicts and leave CPU available for report requests during data load periods. During data load operations, the database remains available for satisfying queries and report requests.


Rosetta@home
Protein Folding, Design, and Docking

UNIVERSITY OF WASHINGTON

Dredd
Active Work Units

Site search [Home] [Join] [About] [Participants] [Community] [Statistics] [login/out]

Results are updated daily. Significantly outlying data may not be displayed in plots.

Active Work Units	Best RMSD user	Best RMSD all users	Best Score user	Best Score all users	# Predictions user	# Predictions all users	Last Updated
• 1hz6A BOINC NATIVEJUMPS CLOSE CHAINBREAKS VARY ALL BOND ANGLES ALL BOND DISTANCES SAVE ALL OUT	0.01	0.01	-148.12	-162.44	31	155413	11-08-06
• 1hz6A BOINC NATIVEJUMPS CLOSE CHAINBREAKS VARY ALL BOND DISTANCES SAVE ALL OUT	0.01	0.01	-140.87	-152.78	55	344997	11-08-06
• 1b72 BOINC POSE ABRELAX VARY ALL BOND ANGLES VARY ALL BOND DISTANCES NEWRELAXFLAGS	0.01	0.01	-126.42	-138.61	36	106332	11-08-06
• DOC 1STF pose u pert bbmin from short relax	0.41	0.35	-307.96	-317.16	45	14792	11-07-06
• DOC 1STF pose bound perturb benchmark	0.23	0.14	-292.75	-293.85	33	16411	11-01-06
• DOC 1CHO pose bound perturb benchmark	0.29	0.19	-248.19	-250.42	46	14500	10-30-06
• DOC 1BRS pose u pert bbmin from better preminimization	1.92	1.73	-221.95	-228.35	58	33671	11-08-06
• 1elwA BOINC FILTERS ABRELAX SAVE ALL OUT frags83	0.64	0.43	-301.2	-307.88	58	110503	11-02-06
• 1r69 LARS ABRELAX PAIR5 BARCODE	1.24	1.02	-141.39	-150.88	180	205457	11-08-06
• 2reb BOINC OLDRELAXFLAGS ABRELAX SAVE ALL OUT	0.76	0.52	-144.26	-149.26	126	359834	11-03-06
• BENCH ABRELAX SAVE ALL OUT 1ctf BARCODE R41 filters	2.83	2.58	-143.52	-152.43	47	30383	11-08-06
• DOC 2KAI U pert bench	1.22	0.94	-234.87	-236.92	34	16808	10-28-06
• 1tiq BOINC ABRELAX SAVE ALL OUT truess	2.16	1.88	-186.3	-195.43	49	19675	10-26-06
• 2reb BOINC POSE ABRELAX NEWRELAXFLAGS	0.85	0.56	-143.9	-160.32	20	89697	11-08-06

Internet 100%

Figure 4. Sample report showing a user's ranked contribution to all experiments

5. Reports and Queries

All reports are rendered on-demand using SQL Server Reporting Services as users request them from the web site. For each report request, Reporting Services issues an SQL query for a data set that is used to generate either tabular text or scatterplot graphic results.

For example, the report shown in Figure 4 uses the following SQL query to report the data contributed to each experiment from a selected user, and the user's best result per experiment in comparison to the *overall* best data point for each experiment:

```
SELECT
   UserID, BatchID, SubBatchName,
   cnt, totcnt,
   u.minrms as thisminrms,
   s.minrms,
   u.minscore as thisminscore,
   s.minscore,
   u.minrms-s.minrms as rmsdiff,
   convert(char(8),lastmodified,10)
                    as lastupdated
FROM RosettaSubBatches b
INNER JOIN RosettaSubBatchUsers u
      on u.subBatchID = b.SubBatchID
INNER JOIN RosettaSubBatchStats s
      on s.SubBatchID = b.SubBatchID
WHERE u.UserID=@UserID
```

A more complex query is necessary to generate the scatterplot from an experiment's results to-date. The plot consists of 2 'layers' – one layer reporting all experimental data points bucketed into a 100 x 100 grid, and the second layer superimposing the data points for the specific requested user, team or host in a different color (See Figure 1.)

The 'background' query, shown in Figure 5, aggregates the detail data into the 100 x 100 buckets, with each bucket identified by its lower bound of RMS and Energy value (*rmshist* and *scorehist* respectively). Parameters include the number of standard deviations above or below the mean to use as the upper limit of each axis (*@stdZoomRms* and *@stdZoomScore*). A column named *theUser* is returned, containing a constant 0, identifying the rows as being identified with all users.

Note that although the query in Figure 5 will return a maximum of 10,000 values, it must scan all rows in ***RosettaResults*** associated with the requested *SubBatchID*, which could total over 1 million rows. But since the default scatter plot settings use the value -2 for *@stdZoomRms* and *@stdZoomScore*, the query above can be rewritten to the following SQL statement referencing the aggregate table for the default scaling:

```
SELECT
   rmshist,
   scorehist,
   cnt,
   0 as theUser
FROM [RosettaResultsAgg_100_-2_-2]
WHERE SubBatchID = @SubBatchID
```

And this query will scan at most 10,000 rows to generate the same result set. The query to populate this aggregate table for a given *SubBatchID*, at data load time, is simply the query in Figure 5 with the appropriate parameter values substituted.

The second query required for the scatterplot returns only those rows associated with the specific UserID to be highlighted in the report. We cannot use preaggregated data for this query since it is UserID-specific, so the SQL statement looks just like the query in Figure 5, except we add UserID = @UserID to the WHERE clause, and use the constant 1 as *theUser*. Although we cannot use a pre-aggregated table for this query, this query will access at most some tens of thousands of rows in the database because of the clustered index present on *SubBatchID* and *UserID*. Typically it executes in only a few milliseconds.

To generate the full scatterplot, we simply UNION ALL the background query with the user-specific query, and sort the results by *theUser*, ensuring that the background rows (*theUser* = 0) are returned before the user-specific rows. We also instruct Reporting Services to color the points based on the value of *theUser*, and vary the darkness of the points based on the value of the counts associated with it. Reporting Services renders points in the order returned from the query, hence the background points will be overlaid by the user-specific points. Note that this layering strategy ensures that any background points associated with the requested user are overwritten by the differently colored user-specific data point, allowing us to use the common aggregate table for the background query at default scaling. A similar strategy is used for the scatterplots that present results by computer or by team.

The scatterplots can be zoomed in and out by adjusting the maximum values on each axis. This in turn varies the parameters *@stdZoomRms* and *@stdZoomScore* passed as query terms inside the URL along with the parameters *@SubBatchName* and *@UserID* when a report is requested.

```
select
  minrms + floor((rms-minrms)/rmsbucket)*rmsbucket as rmshist,
  minscore + floor((score-minscore)/scorebucket)*scorebucket as scorehist,
  count(*) as cnt,
  0 as theUser
from RosettaResults r
inner join
  (select
     SubBatchID,
     minrms,
     stdrms,
     avgrms,
     (avgrms - (@stdZoomRms * stdrms)- minrms)/(100.0) as rmsbucket,
     minscore,
     stdscore,
     avgscore,
     (avgscore - (@stdZoomScore*stdscore) - minscore)/(100.0) as scorebucket,
     totcnt
   from  RosettaSubBatchStats) s
 on s.SubBatchID = r.SubBatchID
 where
   r.SubBatchID = @SubBatchID and
   rms < avgrms - (@stdZoomRms * stdrms) and
   score < avgscore - (@stdZoomScore * stdscore)
 group by
   minrms + floor((rms-minrms)/rmsbucket)*rmsbucket,
   minscore + floor((score-minscore)/scorebucket)*scorebucket,
   totcnt
```

Figure 5. 'Background' portion of query used to generate the scatterplot

6. Results

The design and implementation of this project was done by 2 people in 1 month. It was deployed in the autumn of 2006. By early November 2006, the reporting pages of Rosetta@Home had become one of the most popular sections of the website, generating over 15,000 reports daily. The database holds 115 million results from 3900 active experiments. Ad-hoc reports are rendered and delivered over the Internet with a latency of a few seconds, and the database/reporting server shows only a light CPU load when delivering 1000 reports per hour, offering significant room for growth in concurrent workload. The daily refresh of data completes in less than 1.5 hours, running as a single-threaded process with no adverse impact on reporting availability. During the data load operations, one of the 4 available CPU cores is observed to spike at 100%, leaving at least 75% of total CPU resource available for querying and report rendering.

The reporting system has proven to be an essential tool for Baker lab researchers as well by offering rapid generation and visualization of energy vs. RMSD plots. This provides researchers the opportunity to more easily monitor and make mid-course adjustments to experiments in progress.

There are many opportunities to extend the reporting solution to do even more to encourage community participation and help researchers. For example, time-series reports would be a natural next step -- illustrating the rate of output or level of resource contribution (e.g. experimental results or CPU time contributed per week, per user or team; or aggregate resource consumption over time per experiment.) The lab can also use these reports to recognize or reward accelerated contributions from participants. The reporting system can also be extended as a data analysis tool for researchers. For example, components of the Rosetta score can be saved for analysis, and histograms and overlay plots can be made to compare results from multiple experiments.

7. Conclusions

Adding an on-demand reporting system to Rosetta@Home proved to be a feasible and popular addition to this BOINC-based research project. The use of an integrated platform containing a database, data integration services, and reporting services fostered rapid development of the solution and proved simple to deploy and integrate with the project's existing web site. The techniques used to extract experimental result data from BOINC into a relational database, selectively preaggregate and summarize the data, and invoke participant specific queries that render graphical output are likely to be useful for other distributed computing projects architected atop BOINC or other platforms.

8. References

[Anderson2002] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, Dan Werthimer, "SETI@home: an experiment in public-resource computing," *Communications of the ACM*, Volume 45, Issue 11 (November 2002), pp 56 - 61

[Anderson2004] David P. Anderson, BOINC: a system for public-resource computing and storage, *IEEE/ACM Grid Computing, 2004.*

[Baker2006] David Baker, Proteins by Design, *The Scientist*, July 2006, pp 26 – 32

[Rosetta] http://boinc.bakerlab.org/rosetta/

[SQLServer] http://www.microsoft.com/sql/