2008

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

DDAS – Distributed Data Analysis System

University of North Carolina, Wilmington

MS CSIS Capstone Project Document

Max Rupplin

Summer, 2008

Capstone project submitted as a portion of the requirements for:

Masters of Science Degree in Computer Science and Information Systems

Table of Contents

1 Introduction

1.1 Project Overview

DDAS is an acronym which stands for distributed data analysis system and it is the subject of this paper. Its purpose is to perform all possible linear regressions on otherwise intractably large data sets using the power of desktop grid computing. *Data* sets are *distributed* piecewise to grid clients for computation and after the results have been computed, they are *analyzed* by the *system*. In such a manner large data sets are divided into tractable pieces so that DDAS succeeds in analyzing very large data sets, where serial approaches remain intractable.

DDAS is built on top of a grid computing framework called Berkeley Open Infrastructure for Network Computing, or BOINC. BOINC provides a generic structure on which specific grid implementations may be constructed. It has been used on such notable projects as SETI@home, ClimatePrediction.net and Folding@home. BOINC, as well as DDAS, are discussed in greater detail later in this paper.

1.2 Project Goal

The project's primary goal is to facilitate the analysis of heretofore intractably large statistical data sets. In the case of DDAS, the analysis consists of calculating all possible linear regressions of a data set. Linear regressions often provide useful models for predicting outcomes based on a series of observations. However, calculating every possible regression of a data set is rarely done because of the exponential nature of the computation – each additional observation roughly doubles the computation time.

To get around such limitations, faster probabilistic methods such as stepwise or backwards elimination are often used. These approaches trade ultimate accuracy for speed. They do not check all models but perform a kind of statistical "best effort" to achieve their results. Consequently, while they are faster, they are unable to guarantee the model(s) they return are the

best or even in the same statistical class as the "best tier" models. Thus researchers may find themselves, for want of the best model, coming to the wrong conclusions. DDAS was conceived and implemented to ensure that researchers have the best models from which to base conclusions while at the same time minimizing the costs, temporal or otherwise, of finding those models.

1.3 Project Scope

The current version of DDAS allows researchers, via a web interface, to drop off data sets of up to 128 independent (observational) variables. However, the actual number of columns that the system can handle in a reasonable amount of time is between 30 to 40 independent variables. The reasoning behind this limitation is discussed in section six.

Additionally, the first version will perform only "strict" linear regressions. To clarify, future versions may be implemented to allow the manipulation and appending of observational columns prior to work generation and distribution. That is, new observational values can be derived from existing values via mathematical function(s) such as $\sqrt{x}$, x^2, $\log(x)$, et cetera. These manipulated data could then be appended to the original data set, allowing DDAS to consider and discover non-linear relationship in its models as well as purely linear ones. This will allow potentially greater accuracy in the models.

1.4 Project Benefits

As noted in previous sections, DDAS allows researchers to perform all possible linear regressions on large data sets but, importantly, it does so at no new substantive cost to the university. All of the client hardware used by DDAS has already been purchased by UNCW. The server hardware and software development cost was nominal, approximately $4,000. However, the benefit to cost ratio is probably the single strongest benefit.

According to top500.org, a registrar of the world's fastest supercomputers, the current fastest supercomputer is IBM's Blue Gene Solution. With a total of 212,992 processors and 73,728 GB

of RAM, its computational power peaks at 596.378 teraFLOPS. Its cost, according to several websites, was on the order of $100,000,000. To put the potential power of DDAS in perspective, the combined computation of all major known BOINC projects is estimated to average over 900 teraFLOPS - nearly twice that of the fastest supercomputer. SETI@Home alone averages 265 teraFLOPS - more power than the second most powerful supercomputer in the world.

To summarize, as more computers are added to the system, internally or from extra-university volunteers, the benefit increases but the cost remains constant, as the hardware is a sunk cost. Finally, as the computer infrastructure of UNCW is renewed over time, so is the computational infrastructure of DDAS. This, in turn, means that DDAS will get faster as it gets older, instead of becoming irrelevant. (1) (2) (3)

2 BOINC - Berkeley Open Infrastructure Network Computing

2.1 Overview

BOINC is a free, mature, open source implementation of the grid desktop computing paradigm. It provides the means for massive parallelization of problems, using for computation geographically disperse heterogeneous desktop class computers rather than relying on the brawn and consequent high cost of a typical supercomputer. BOINC therefore positions itself as a low cost and powerful alternative to other computing solutions. (4)

BOINC uses a client-server model. Projects typically consist of one or more physical servers, to house the requisite back-end components, and one or more volunteer client desktop machines, to provide the computational power. A BOINC server consists of a mix of generic components and project specific, researcher defined and implemented components. The initial work of the researcher, then, is to define and implement the project specific components, given the goals of the project and constraints of BOINC.

2.2 BOINC Background

BOINC was originally developed by UC Berkeley in the mid-1990s to make use of the increasingly ubiquitous desktop computer and its idle processor. Among the project's sociological goals were that it should attract public interest and that it should be aesthetically pleasing to end users. As a result of this thinking, SETI@Home and its screensaver were born. BOINC has since been used on dozens of notable problems.

BOINC was not originally intended to be the generic infrastructure it is today, but rather a kind of experiment in the then emerging field of grid desktop computing. Over time the SETI project was made more general and its weaknesses, such as security and fraud, were addressed. (4) (5)

2.3 Typical Users

BOINC is typically a volunteer driven system. Typically computation occurs on the processors of geographically disperse project volunteers; however, universities or other large organizations may provide some, or all of the processing power for a project.

Project volunteers may attach to or leave a project at any time. Volunteers may enlist in and perform computation for multiple projects concurrently. Volunteers are usually rewarded credit as incentive for computation performed on a per-project basis. Groups of volunteers may form teams for competition or merely for fun.

2.4 Appropriate Projects

Due to the nature of BOINC's grid implementation, only certain types of parallelizable applications are suitable. If a problem's space can be divided into multiple, relatively small units and the computation of a single unit does not need immediate access to the data, memory, and/or results of its fellow units, it is most likely suitable. Conversely, problems which require concurrent access to the data, memory and/or results of other problem units are probably not suitable. Because DDAS' data set may be divided into smaller tasks, relative to the whole, and these tasks can be calculated independently from each other, it is suitable to be implemented using BOINC. (6)

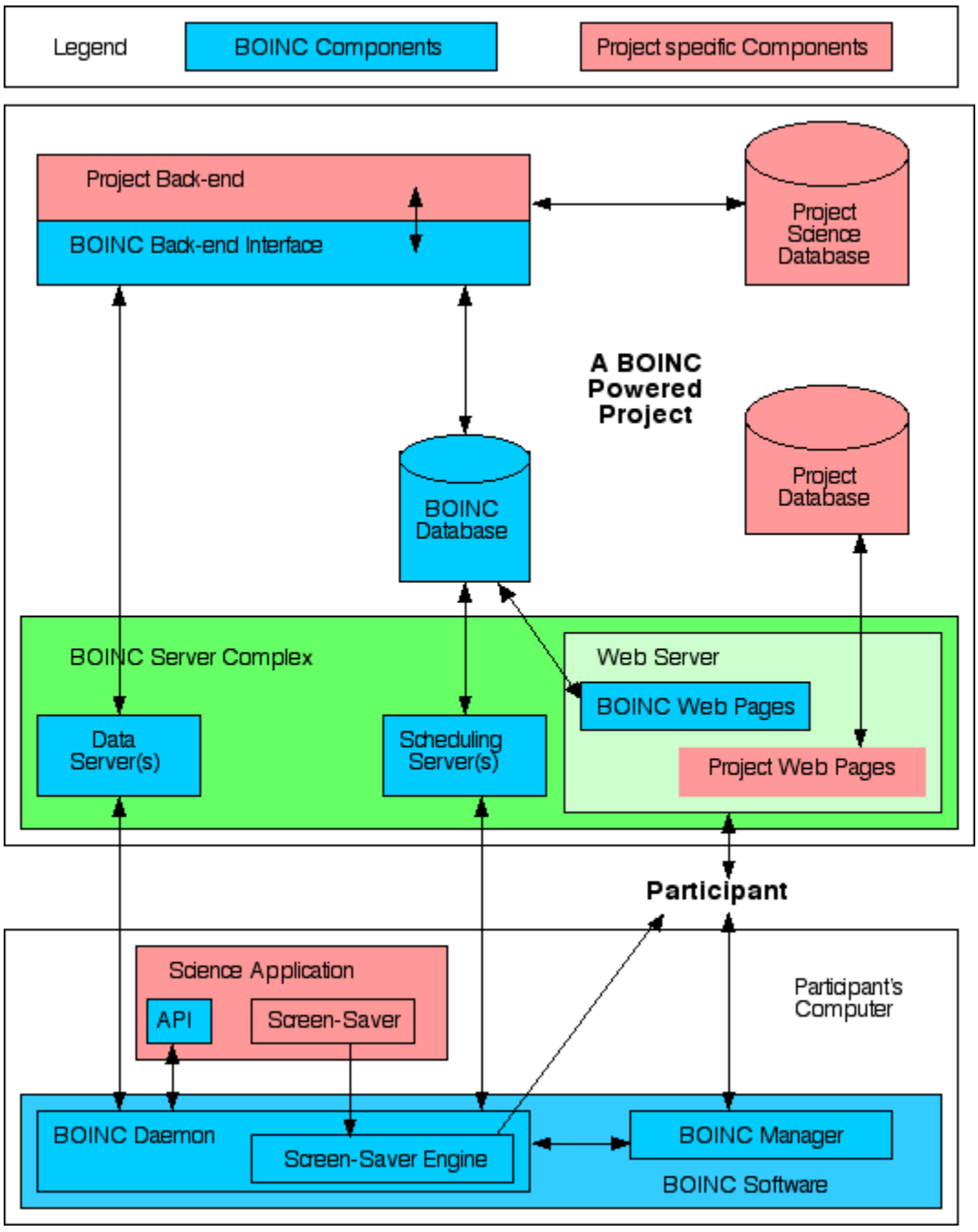
Pre-existing applications, written in C, C++ or FORTRAN may be used with little or no modification with BOINC.

2.5 BOINC Architecture

BOINC's architecture is based on a client-server model. Each project may have one or more physical servers (for work scheduling, data warehousing, etc.) or, alternately, a single physical server may house multiple, distinct projects. In order to interface with a project or projects,

clients must download an executable referred to as the BOINC manager. The manager acts as the communication mediator between the client and the server(s). Clients sign up for (attach to) one or more projects and are given work based on their computer's capabilities. The client computers interface with the BOINC manager, which in turn interfaces with the project's server. See Figure 3.1 for an overview of BOINC's components.

Figure 3.1 – BOINC Architectural Components



3 All Possible Combinations Linear Regression

3.1 Overview

The purpose of DDAS is to isolate the best possible linear model with respect to a system of observations and a set of outcomes. This is achieved through the checking of all possible combinations of linear regressions. This means checking every possible model derived by completely iterating through the set of possible combinations of independent columns.

3.2 Linear Regressions

Linear regression is a statistical method which can be used to predict outcomes based on a series of observations. These are termed dependent and independent variables respectively. The output of a linear regression is a predictive model in the form a linear equation. The resulting model consists of a y-intercept and a series of coefficients. The coefficients serve to linearly scale inputs based on their relative importance and sign.

For example, take the following hypothetical data set which has as its dependent variable the person's weight:

Name	Gender	Age	Height	Weight
Joe	1	25	72	178
Jill	0	32	68	122
Jack	1	27	69	167
John	1	45	67	210
Jane	0	38	62	108

When the data set is run through a linear regression the following model is returned:

$$\text{Weight} = -337.9447029 + 54.63010021 * \text{Gender} + 2.965282749 * \text{Age} + 5.371689334 * \text{Height}$$

It can be seen that gender is the dominant input variable, while height is the second most dominant. Below is the output of the predictor model. Note the estimated weights are reasonably close to the actual weights.

Name	Gender	Age	Height	Weight	Weight Estimate
Joe	1	25	72	178	177.5790981
Jill	0	32	68	122	122.2192198
Jack	1	27	69	167	167.3945956
John	1	45	67	210	210.0263064
Jane	0	38	62	108	107.7807802

3.3 Benefits

3.3.1 Guarantee of Quality

Performing all possible combinations of linear regressions means performing linear regressions on all possible combinations of independent columns. This approach, as opposed to heuristic approaches like stepwise or backward elimination, guarantees the best possible linear model. This is important because, in many cases, large quantities of work can be based on such a predictive model. If the conclusions contained therein are based on a flawed model, subsequent research based on those conclusions will also be based on less than ideal models.

3.3.2 Ease of Use

Linear regressions provide a foundation for researchers that may serve as a basis for further research. While non-linear approaches may yield more accurate models, they are more complicated both to implement and interpret. Linear regressions, on the other hand, are relatively simple to implement and interpret.

3.3.3 An Initial Basis

When a new data set is obtained by a researcher, the researcher may not know what type of model best explains the data. There may be linear relationships for certain variables while other relationships may be more complex or subtle. Given this situation a linear model may be used as a first effort at a predictive model. While it may not always provide the most accurate models, linear regressions allow the researcher to glean some idea of the basic correlations between the independent and dependent variables. This model may be useful in and of itself, or in facilitating the creation of other, more complicated models.

3.4 Model Building

3.4.1 The Real World

A model, in the sense the word is used herein, is a mathematical description of the relationship of one or more input variables to a single output variable. Models provide a mathematical correlation between inputs and outputs.

Because the physical universe is constantly in flux and imperfectly measurable, all mathematical models are apt to contain some kind of error. That is, it is extremely unlikely that a model will perfectly predict real world phenomenon, but rather serves to estimate the real world.

3.4.2 Parsimony

Occam's razor states that the explanation of any phenomenon should make as few assumptions as possible eliminating those that make no difference in the observable predictions. In terms of linear regressions this means that if two or more models have roughly the same level of predictive quality, one ought to choose the model with the most concise mathematical description; that is with the fewest coefficients.

3.4.3 Non-Linearity

The problem of creating a linear model is deciding which independent variables affect the outcome variable and how. However some systems are non-linear in nature and therefore efforts to model those systems using linear models will produce poor results. As a hypothetical example, consider the growth of bacteria. Each bacterium creates two copies of itself and these copies in turn create copies of themselves. This is an example of an exponential growth system – a type of system that linear approaches may poorly model.

While linear regressions do not typically allow one to model non-linear systems well, there is a clever solution. By modifying the columns using non-linear equations such as x^2,$\sqrt{x}$ etc., the system is permitted to consider some possible non-linear relationships. The new columns are appended to the original ones. While this benefit is nice, it does come at a high cost. Each non-linear addition results in a multiplicative increase in the number of columns. That is, including the x^2 terms would increase the column count by 100% and including both x^2 and$\sqrt{x}$ would increase the column count by 200%. Furthermore each additional column roughly increases the amount of work by a factor of two.

3.4.4 Model Selection Methods

Because it requires exponential time to compute all possible combinations other, faster methods were devised. Stepwise linear regressions and backward elimination are the two most frequently used substitutes. In essence stepwise starts with no columns and adds columns to that empty set using combinations of linear, surface and univariate terms. Backwards elimination works in reverse; it starts with all the independent variables and works to eliminate variables below some threshold of significance. The trade-off is time versus the quality of the model. Stepwise and backward elimination are faster yet provide no guarantee for the quality of their results; all possible combinations guarantee the best model given the same data set as input but takes exponential time to compute with respect to the number of columns. (7) (8)

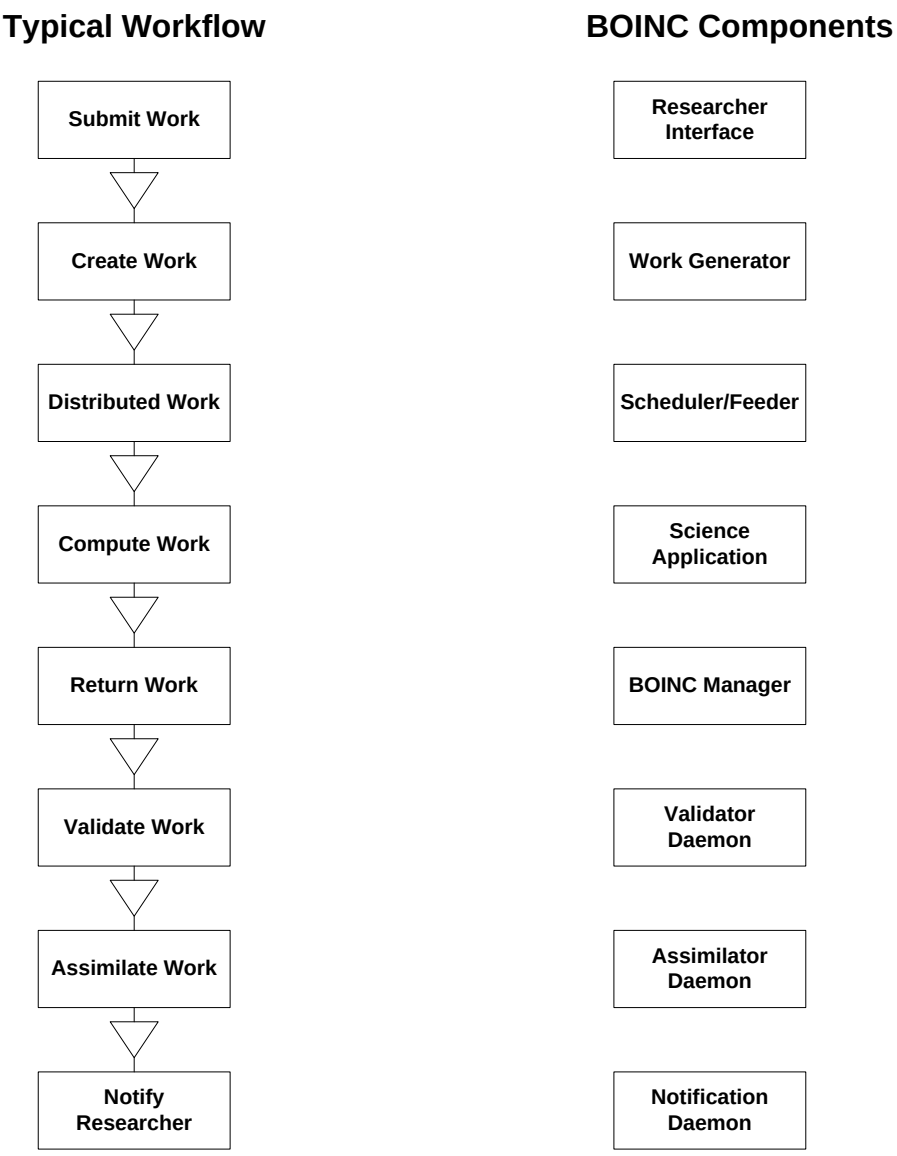
4 DDAS – Distributed Data Analysis System

4.1 Overview

The purpose of DDAS is to perform all linear regressions on large data sets, giving the researcher the best linear model for predicting some outcome. To clarify, every possible combination of the observational columns is used as an input to the linear regressions. Each combination returns, among other things, the sum of squared error (SSE), and the multiplicative coefficients associated with their respective column. These coefficients, tied to their respective columns, become the linear equations which provide the ability to predict system behavior. As all possible combinations are computed, the models with the lowest SSEs (the best predictors) are kept, while models with SSEs above some quantity are discarded.

DDAS consists of several BOINC components. Some of these components reside on the server and some on the client. Some of the components are generic, (the same across all BOINC projects) like the scheduler, and some are project specific, like the DDAS validator. The constituent components are explored in greater detail in this section. To facilitate understanding, Figure 4.1, a workflow diagram which maps DDAS' components against their more general responsibilities, is included below. The key generic BOINC components, not discussed in this section, are defined in the glossary found in the appendix.

Figure 4.1 - DDAS Workflow Diagram



4.2 DDAS Server Components

BOINC is based on a client-server model. From the perspective of the server, certain tasks must be performed periodically in order to have a running project. These tasks map onto components. BOINC creates and manages certain universal components while certain project specific components must be implemented on a per-project basis. Here only the project specific components are discussed in this section. See Figure 3.1 for a stronger delineation between BOINC and project specific components. The first server component for DDAS is the web interface, allowing researchers to drop off work.

After a researcher submits a job, the server must break it up into small pieces suitable for distribution to clients. This portion of the cycle is performed by the work generator component. A hypothetical data set with 16 independent variables has 2^{16}-1 or 65,535 combinations. The data set may be divided in any subset of independent variables. But to facilitate this example, assume the ideal size of client work unit is approximately 10,000 regressions. That is, each unit of work is compromised of 10,000 unique linear regressions. The first work unit is responsible for combinations one to 10,000. The second work unit is responsible for combinations 10,001 through 20,000 and so on, until all 65,535 combinations have been assigned.

The practical way of ensuring that all regressions are computed and that they are unique is to map the current iteration onto its binary value. The value representing the current regression combination is converted into a string of ones and zeros. The subset of columns with one for its value when grouped together become the input columns (the observational columns to be considered) for that particular regression. Each column with zero for that iteration's binary value is withheld from that particular regression.

Once the work has been generated, it is distributed to clients. The clients compute their portion of the data set and return a certain number of best models. The researcher may specify the number of models he or she wishes returned. This is colloquially referred to as the "top n" results. Once the server starts to receive these results back from clients, the third project specific

component takes over. This component determines whether results received from clients are valid, and is termed the validator. Typically duplicate, redundant work units are distributed for computation in order to facilitate validation and to prevent error and/or fraud.

For DDAS, the linear models returned by clients are compared against one another for validity. More specifically, if the regression coefficients for a given model are sufficiently similar to the model(s) returned from other work units, that model is considered valid and is labeled as valid. A client may return one or more models from its computation. If all models for a given work unit agree, the entire work unit is considered correct and is then ready to be assimilated by the system in some fashion. If any part of the result is in error, the entire result is discarded and new work units may be created and distributed as necessary.

Handling result assimilation is the final major project specific server-side component. Once a work unit has been validated, the assimilator must decide what to do with it. In the case of DDAS, the combination number (the string representation of a binary number), the SSE, the raw XML file returned by the client and the associated regression coefficients are stored on a per model basis.

As more results are assimilated an optional skimming process occurs. Only a certain number of results, the ones with the lowest SSEs, are kept. This is to save space in the database which may become full with old and/or irrelevant data. Once all the results for a job have been assimilated the researcher is notified via e-mail of this fact and may download the best models for that particular job.

4.3 DDAS Client Components

Up to this point the discussion has been only of server-side components; however, some attention must also be given to the client portion of the system. There are fewer client side components than server ones, but as with their server side counterparts, some are project specific and some are not. The project independent components are discussed in the BOINC

Architecture section of this paper. The science application is the primary project specific component.

The science application defines what clients do as volunteers for the project. In the case of DDAS, the clients perform the actual linear regressions and so need to have access to the requisite math libraries. However the science application must also interface with BOINC in order to do such things as automatically upload results or download new ones. Interfacing with BOINC is accomplished with BOINC API calls and is fairly straightforward. Finally, the client typically interfaces with a screensaver which may provide some functionality like a progress indicator.

When clients attach to a BOINC powered project, they receive work from the project server(s). As part of this work, the client must download an executable file – the project specific application. In the case of DDAS this is the code that handles calculating regressions and logistics like iterating over a portion of the data set. Clients also typically download a data file containing a subset (all or a portion) of the problem. The executable reads the data file and begins to perform its work.

The actual process is slightly different with DDAS. While the client downloads the executable and data file it also downloads result, scoping and state files. The result file is the final destination of the client computation; when the client is done it writes its data here. The scoping file tells the client where to begin and where to end with respect to the totality of combinations. It also contains data to initialize the client. Finally, the state file is a placeholder for progress. If the client is interrupted mid-computation, all the work will not be lost as it is stored in the state file periodically.

5 DDAS Implementation

5.1 Project Hypotheses

- DDAS provides linear regression results as accurate and as quickly as SAS.
- DDAS provides the ability to compute all possible combinations of linear regressions on heretofore intractably large data sets.

5.2 Hypotheses Revisited

Based on the iris data set it may be observed that SAS and DDAS return, to a high degree of precision, essentially the same results. See Table 3 in section 9.4.3 for the data returned by SAS and DDAS. More data sets should be checked to ensure results are consistent. However, for reasons not yet understood, SAS was much faster than DDAS. This is likely due to the fact that SAS uses only full rank matrices when computing regressions. It is currently unknown whether this approach is heuristic in nature. However, SAS provides results orders of magnitude more quickly than the pure brute-force approach used by DDAS. Understanding what SAS is doing and if that process can be adapted to DDAS are areas of future research.

Because DDAS does provide so much computational power it will indeed be able to compute all possible combinations of regression more quickly than a single computer. With 64 computers it can do a year's work in less than a week. However, because of the exponential relationship between the execution time and number of columns of a data set, a standard brute force approach will always be limited by the size of the input. Currently, a data set with 32 columns and 768 rows takes on the order of ten days with a grid of 64 computers. The computers are primarily 3.2 GHz Pentium IV CPUs with hyper-threading running Windows XP. Because this is not very many columns, future work should be conducted to determine if the SAS approach of only considering matrices of full rank would be worth implementing in DDAS.

Logically checking every possible combination of model given a set of independent columns will guarantee the best possible linear model. However, because heuristic approaches like stepwise may use surface variables and univariate scalings, if all possible regressions do not explicitly consider these possibilities then it can no longer guarantee the best result. Since DDAS currently does not offer the ability to automatically append these variations and creating and appending them manually can be tedious, the heuristic approach may be more appealing. This is an area of future work for DDAS.

5.3 Current Status

5.3.1 Server Components

Web Interface

The first server component in terms of interacting with the system is the web interface. Much of the initial framework for the web interface was implemented by Royce Nobles, a fellow graduate student at UNCW.

Currently, researchers are able to log into the system. Once they are logged in they may submit new work, check the status of existing work and download the results of any completed jobs. Researchers are able to specify the number of models they would like in their final report as well as the desired quorum size on a per job basis. The quorum size is a simple way to detect and prevent error and/or fraud originating from participant computers by duplicating work. See Quorum of Results in the Definitions sections of the appendix for more information.

The functions which remain for the web interface are all non-vital. For instance, a dynamic, database driven page could show data about the project. This would be useful for attracting potential volunteers and/or researchers. The page could show data such as how many volunteers are attached to the project, how many floating point operations per second (FLOPS) the project is able to provide, et cetera. Future versions of the web interface should give the researcher even

greater control over their projects. The researcher should be able to pause one project and start another. While submitting a new job the researcher should be able to specify if it should run in parallel or in serial with his other jobs. These types of controls would add a nice finishing touch to the interface.

Another important function, touched on in other parts of this paper, would be the creation of a built-in data set manipulation system. Researchers, via the web interface, could specify that new observational values derived from existing values be appended to their original data set. See the second paragraph of section 1.3 for further details.

Finally, an administrative interface was envisioned but never implemented. This would allow the project administrator to log in and view the system and make changes as necessary. This, however, is left for future implementers.

Work Generator

The work generator has been working properly for some time now. However, it had certain properties which made it less than ideal. First, the generator allocated all its memory in bulk at the beginning of the process. The amount of memory was always the same amount. This created the problem of consuming a potentially large and unnecessary amount of memory while also failing to ensure that enough memory was allocated for a very large project. The second fatal flaw of the first version of the work generator was that it created all the work at once. The problem with this is that it totally consumed the processor. Additionally, the number of work units created could exceed the available disk space in both the database and the file system.

Since DDAS was envisioned as a means to tackle very large problems, it seemed counterintuitive to leave the work generator in this state. Consequently, the two main flaws were fixed. First, the amount of memory is now dynamically allocated, which is to say the amount depends on the number of work units to be created. Secondly, the number of work units created at once has been limited. This prevents the system from freezing under the circumstance that a very large

data set is dropped off. It also gives the BOINC file deletion component time to delete old work units from the database and file system before new units are made. The final obvious benefit of limiting the number of work units created at once is that it creates the effect of serialization across researcher jobs. One researcher does not have to wait a month for a previous job to complete before his or her new work receives attention.

Validator

The validator currently has two points of validation. First it takes two results and compares the number of models contained therein. If the number of models agrees, then each model's regression data (its SSE and coefficients) are compared against one another. If these data agree within a tolerance of one percent, the results are both considered valid. If this check fails at any point neither result is considered valid and more results may be sent out to clients to resolve the disagreement. Currently if results vary by less than one percent, they are considered to be the same. One percent tolerance was chosen initially as it seemed to work well. No mathematical justification is offered herein and such work is fodder for the future work section. No future work on the validator itself is currently planned but erroneous results should be fed into the validator to verify that negatives are discarded as well as that positives are kept. The validator is in production phase and has been commented using Doxygen.

Assimilator

The assimilator periodically checks if a work unit has a canonical result. If a canonical result exists then that work unit has been validated. At this point the assimilator inserts mapping data, the output file, the data rows and the coefficients of the associated canonical result into the science database. The mapping data provides a quick mapping between a job, a work unit and its canonical result. The output file is the XML file returned by the science application. An example may be found in the appendix. The data contained therein – the SSEs, the definition number, the binary representation and the coefficients are all stored in the science database. There they remain until the individual results are reconstructed for the researcher, in the form of

an XML document, available for download. The assimilator is in production phase and has been commented using Doxygen. No future work on this component is currently planned.

Researcher Notification Mechanism

The researcher notification mechanism is an automated e-mail daemon. Periodically (currently every 24 hours) researchers are e-mailed the statuses of their jobs. Also, when jobs are completed and submitted researchers are sent e-mails to that effect.

What remains is the task of consolidating the statuses of many jobs into a single e-mail. Currently researchers receive one e-mail per submitted job. Additionally, e-mail should be forwarded to the proper UNCW server, smtp.uncw.edu, for extra-university deliveries. Currently notifications are limited to e-mail addresses belonging to UNCW.

5.3.2 Client Components

Science Application

The science application has undergone numerous revisions since the beginning of the project. In addition to the basic ability to perform regressions two new features have been added to the science application. A checkpointing feature has been added; this allows the client computer to save the best current results periodically. This means that even if the machine is interrupted or powered down abruptly, the best results thus far and an iteration number are saved as a future starting point. Secondly, it was discovered that the data sets were missing an implied y-intercept column. This has been corrected. No new features are currently planned.

Client Screensaver

The current client screensaver was only slightly modified from the original, generic one that came with BOINC. A UNCW logo replaces the BOINC logo. A progress bar in the colors of UNCW has been added and the text now moves horizontally back and forth to avoid burning in their characters.

6 System Analysis

6.1 Work Unit Size

Tasks, otherwise known as results, are the basic unit of computation for clients using BOINC. Their size is defined by the project programmer(s). They can be very small in scope, requiring less than a minute of client computation or they can be arbitrarily large. The system programmer, then, is faced with the problem of choosing the optimal size for a task. A very large task may minimize network traffic but the researcher may have to wait until all, or nearly all of the data set has been computed to see a single result. Additionally, if a work unit is very large, much time may be wasted if the client's result is erroneous. A very small task size, on the other hand, provides the benefit of returning answers very quickly but as the cost of more network traffic and a busier server complex.

Analytically the goal is to minimize both the amount of idle CPU time of project clients and the total network bandwidth consumed. Therefore, as a first guideline, the number of tasks should always be equal to or greater than the number of CPUs available. Newer computers may have two, four or more CPUs. Fewer tasks would necessarily leave one or more CPUs idle and therefore negatively impact the efficiency of the system. Because most jobs are supposed to be very large (the purpose of DDAS is to distribute large data sets), having a job divided into very few tasks would most likely not make sense.

Since DDAS will primarily rely upon UNCW computers, and the UNCW intranet is very high speed, bandwidth is not a large concern. This means jobs can be very small or large without being strongly affected by the underlying network infrastructure. One, however, should not neglect considering non-university volunteers. A task should be small enough that even low bandwidth computers can download it within minutes but large enough to be meaningful. A typical task, then, should require not more than one or two hundred kilobytes of data transfer.

Because checkpointing has been implemented on the clients, an interruption of the computation does not equate to a loss of data. This means that small task sizes no longer have the inherent benefit of protecting against data loss via their brief computational windows.

Currently the task size is derived by considering three values: the number of rows in the data set, the number of iterations required by the data set ($2^{\#\ independent\ columns} - 1$) and a "magic" number, a.k.a. the "work unit dimension". In the case of DDAS this magic number is 500,000,000. It was derived from observing the time required to complete a task, given its size, with the goal of an average time being between 30 and 90 minutes.

A task is assigned a certain subset of iterations based on the magic number divided by the number of rows in the data set. For example, consider a data set with 1000 rows and 32 independent columns. Each task would be responsible for 500,000 iterations given that the data set has 1000 rows. The number of iterations (the scope of a task) then, changes inversely with respect to a change in the number of data rows.

As noted above, tasks are intended to require 30 to 90 minutes of computation each, depending on the speed of the computer. This is enough work to be meaningful, and gets the results back to the server in a fairly rapid fashion. The impact of fraudulent and/or erroneous results is mitigated by the relative brevity of computation. This brevity also benefits researchers who may wish to see their best result(s) periodically, instead of waiting until the very end of the job. Since many of the regression results may be very similar in terms of quality, this might be of some benefit to researchers.

Also, a task size of this magnitude also ensures that the number of tasks generated is large enough to always have work for non-university volunteer computers. A 32 column data set with 1000 rows would generate 8,400 work units. Each work unit would have one or more associated tasks. Thus, even with 8,400 volunteer CPUs, the DDAS server will yet have work to distribute.

As a final justification for this size, consider Moore's law. It states that approximately every two years the number of transistors on a CPU doubles. This is often used, rightly or wrongly, to express that speed also doubles every two years. Assuming that this is true, in two years a task would take half as long to compute as it does today. In four year it would take a quarter of the time, et cetera. Based on empirical data, a 3.2 GHz Pentium 4 with hyper-threading takes roughly an hour to complete a task as described above. Thus even in four years, the current magic number will still be sufficient to generate meaningful tasks, and yet small enough that older machines will still be able to contribute. The primary benefit of this design is that the researcher only needs to alter a single variable to alter the amount of work per task.

6.2 Breakeven vs. Single Computer

Again, for clarity, a quick review of some terminology: A project has one or more jobs – these are what researchers submit for analysis. A job is divided into a certain number of work units. Work units serve as templates for tasks. A work unit has one or more associated tasks; tasks are instances of work units. To be clear, clients perform computations on tasks, *not* the work units.

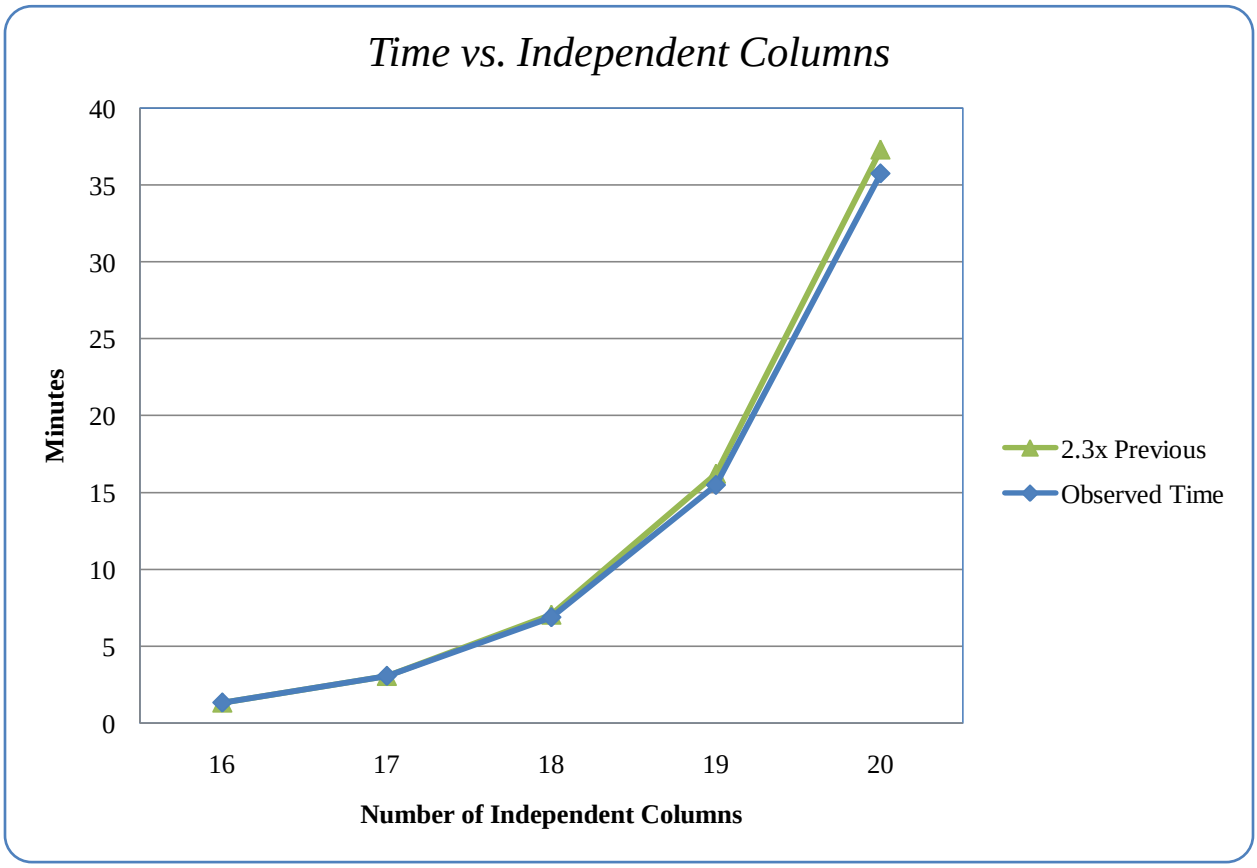
There are four variables which can be used as inputs for a function that predicts how long a certain job will take to compute. The first and the computationally most significant is the number of rows in the data set. Second in importance is the number of independent columns in the data set. Third is the size of the quorum. Finally, the last variable is the number of computers available to perform work.

For example, consider a quorum size of two. This would mean that for a given work unit there must be two tasks computed by the grid before any validation can be done for that particular work unit. The quorum size then serves as a simple multiplier for the number of tasks (the benefit of this redundancy is protection against fraud and/or erroneous results). A quorum size of three would mean that every work unit for that project has three associated tasks. Consequently, 100 hours of work for a single computer becomes 300 hours of tasks for the grid to compute.

Everything else equal, if the number of computers available to the grid is greater than the quorum size, DDAS is typically faster than a single computer, *for large projects*. There is, of course, logistical overhead to the DDAS system. The work must be divided, distributed, validated and assimilated. As a very rough (and pessimistic) estimate, consider it takes an hour, in total, to provide these services. In reality, these services are a function of the size of the job, but for simplicity we make this assumption.

Another factor is the number of computers available in the grid. The number of computers in the grid serves to mitigate potential increases in the number of columns, the number of rows, the quorum size or any combination of the three. Empirical observation shows that appending a new column increases the total work time by a factor of approximately 2.3. The computation time is increased by a factor of approximately 3.0 for each doubling of the number of data rows. To offset the addition of a column to the data set, the number of computers would have to increase by approximately 2.3 for the runtime to be approximately the same.

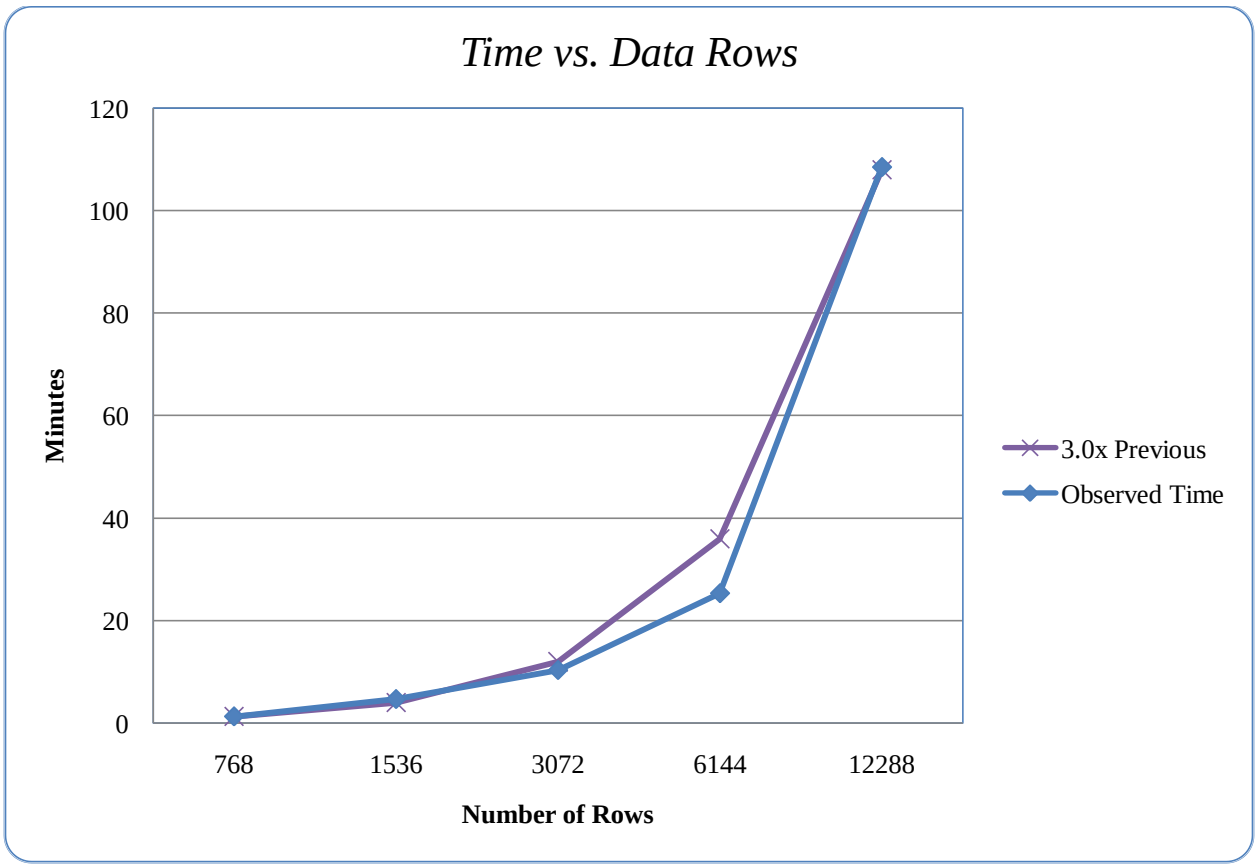
Figure 6.1 – Time vs. Number of Independent Columns (Single Computer)



The two other factors considered are the number of rows and columns in the data set. Figure 6.1 shows that a comprehensive 16 column regression with 1024 rows would take approximately one minute on a fast computer using the DDAS client. An hour's worth of work could be created by increasing the number of columns from 16 to 21. Alternatively, the number of rows might also be increased. According to Figure 6.2, it would take approximately 8000 rows to create an hour's work, if the number of columns remained constant.

Given these graphs as rough guides, equations can be created to estimate the amount of time a data set will take to complete. Below are the derived equations:

Figure 6.2 – Time vs. Number of Data Rows (Single Computer)



The Row Equation

- Find $\log_2$ of the number of data rows in the data set: $\log_2(5000) = 12.29$
- Find $\log_2$ of a known amount of work, in this case 1024 rows: $\log_2(1024) = 10$
- Subtract the two: $12.29 - 10 = 2.29$
- Use this value as the exponent: $3^{2.29} = 12.38$
- So, 5000 rows require 12.38 as much time as 1024

The Column Equation

- Find of the number of data columns in the data set: 26 columns
- Find a known amount of work: 16 columns = 1 minute
- Subtract the two: 26 – 16 = 10
- Take this value as your exponent: $2.3^{10} = 4142.65$
- So, 26 columns require 4142.65 as much time as 16

The Quorum Effect

As noted earlier, the quorum size is a simple multiplication operation. If the quorum size is doubled from two to four, the amount of work and therefore time to completion also doubles.

The Grid Size Effect

The more computers the grid has, the more work can be done per unit time. A rule of thumb would say that for every doubling of the number of computers available, the time to completion would be halved. Linear speedup is expected because of the independent tasks of computation for DDAS as mentioned in section 2.4. Similarly, the number of computers would have to increase by a factor of 2.3 (see Figure 6.1) to counterbalance the effect on adding an additional column to the data set.

The Unifying Equation

The unifying equation is simply the result of combining the row, column and quorum equations into a single equation. For a single computer it looks like:

$$Total\ Time = Base\ Work\ Unit\ Time \times Row\ Equation \times Column\ Equation \times Quorum\ Size$$

The *base work unit*, in this example, is a data set with 16 independent columns and 1024 rows. The *base work unit time* is how long it takes for a single computer to compute all regressions for the base work unit, in this case, approximately one minute. The base units were chosen merely

to facilitate the math, and are otherwise arbitrary. Assuming a hypothetical data set has 5000 rows, 26 columns and a quorum size of two, the unifying equation would be:

$$102{,}572\ minutes \approx 1\ minute \times 12.38\ \times 4142.65\ \times 2$$

$$\downarrow$$

$$102{,}572\ minutes \approx 71\ days$$

For a single computer this data set would take roughly 71 days to compute. However, since our grid has 64 computers, we can divide this work equally among them. The work per computer becomes:

$$\frac{71\ days\ of\ work}{64\ computers} \approx \frac{1.1\ days\ of\ work}{computer}$$

6.3 Lower Practical Bounds of DDAS

Given these equations the question remains at what point should one use DDAS instead of a single machine. Assuming naively that the logistics of DDAS take an hour, any data set requiring less than an hour of time will *always* take longer on DDAS than a single machine.

Taking an equally practical view of things, assume that the longest a researcher typically would be willing to have his personal computer occupied by a regression computation would be overnight. Assuming normal business hours, the researcher's computer has approximately fifteen hours of uninterrupted computation time. Any data set requiring more than fifteen hours of attention, then, would be a candidate for DDAS.

6.4 Upper Bounds of DDAS

The lower limit of computational feasibility has been established. Any data set requiring less than one or two hours of serial computation will always take longer on DDAS than on a single computer and any data set requiring less than ten hours of computation could simply be done overnight. But what are the theoretical upper limits of DDAS?

In the section addressing the unifying equation, a base work unit was postulated to exist. The purpose of the base work unit is to serve as a means to aid extrapolation. It took one minute for a fast machine to compute this base work unit.

The most important variables in determining the duration of a job are the number of columns and the number of rows. The quorum size and the number of computers also play a role, but their effect is much easier to understand. First let us examine the scenario where the number of columns is varied within the unifying equation.

Between the definition of the base work unit and the unifying equation, one has the means to map the relationship between a data set of some dimension and the resulting amount of work, in terms of actual time. This allows one to discover the limits of DDAS given some amount of time above which the wait is considered "too long". One should then define what is meant by "too long". Since a year is a good yardstick for further extrapolation and it is also a long time to wait for a result, it is a suitable candidate to define what is meant by intractable.

The Independent Variable Limit

To find the upper limit of the number of columns one increases the number of columns while holding all other variables constant and observes the predicted duration. Since the number of columns is the only variable changing (with respect to the base work unit), all variables, with the exception of the quorum size and the number of computers, are assumed to be one. The unifying equation for a year's work for 64 computers and a quorum size of two looks like:

$$(365\ days\ \times 64\ computers\ \div 2\ quorum\ size\ ) \approx (2.3^{20}\ minutes)$$

$$\downarrow$$

$$(525600\ minutes\ \times 64\ computers\ \div 2\ quorum\ size\ ) \approx (2.3^{20}\ minutes)$$

Recall that the base work unit is defined as having 16 columns and 1024 rows. What this result says is that one may add 20 more columns (for a total of 36 columns with 1024 rows) to the base work unit before the 64 computers would have roughly a year's work.

The Data Row Limit

An unexpected result from the analysis in this section is the observation that doubling the number of rows in the data set has a larger impact than adding one column. Because adding a bit to the columns increases the amount of computation by a factor of two, the number of data rows was also doubled, to create the effect of doubling in the vertical direction. This is meant to give some equality and meaning to any comparisons between columns and rows for a data set. See Figure 6.2 for the effect of doubling the data rows.

Recall from the row equation that each doubling of the number of rows sees roughly a tripling of the time to complete the work on that data set. The unifying equation for a year's worth of work, only modifying the number of rows of the base work unit looks like:

$$(365\ days\ \times 64\ computers\ \div 2\ quorum\ size\ ) \approx (3^{15}\ minutes)$$

$$\downarrow$$

$$(525600\ minutes\ \times 64\ computers\ \div 2\ quorum\ size\ ) \approx (3^{15}\ minutes)$$

This result indicates the number of rows may be increased by a factor of 2^{15}, for a total of $2^{10} \times 2^{15} = 2^{25} = 33{,}554{,}432$ rows.

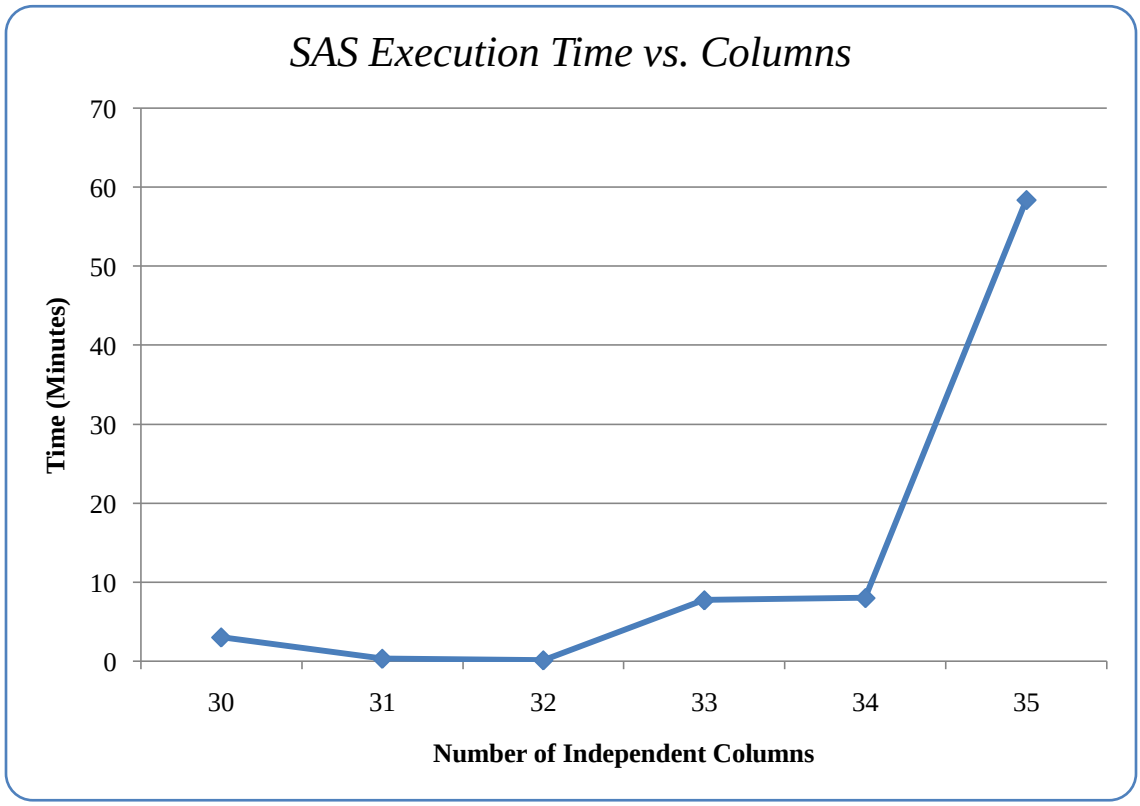
6.5 DDAS versus SAS

To validate DDAS the same data sets were run through SAS as DDAS. The results were compared for accuracy and an example comparison may be found in the tables section of the appendix. In all cases DDAS provided results as accurate as those produced by SAS. However, SAS displayed some unexpected behavior: SAS handles all possible regressions using only matrices of full rank.

The result of this behavior is that SAS only actually considers a fraction of the entirety of possible model combinations. As a result, it is able to handle very large data sets, on the order of 35 independent columns in about an hour. Again, this is because SAS is not actually checking each possible model iteratively. Whether this necessarily means that SAS will only occasionally find the best model or whether this is simply a more clever way to approach the problem is unclear.

Because DDAS and SAS do not perform the same way, no real comparison between the systems in terms of speed is made herein; however, Figure 7.1 shows the speed of SAS with what would be considered a large data set for DDAS. Compare this with Figure 6.1, the graph of DDAS's execution time and note the number of independent columns for each. Because the approach used by SAS to calculate all possible regressions may derive its speed from a heuristic approach and that it is known that DDAS does provide a non-heuristic approach, there is still reason to consider using DDAS; it guarantees the best result. Even if the approach used by SAS is non-heuristic in nature, future versions of DDAS may well be able to adapt to use the same approach.

Figure 6.3 – (SAS) Time vs. Number of Independent Columns



7 Summary and Conclusions

7.1 The Benefits

There are many benefits of DDAS. Because DDAS is not specific to one area of research, any area of research which can benefit from predictive models may benefit from DDAS. DDAS puts the power of a statistical supercomputer in reach of university researchers and gives them an intuitive and simple interface to the system. DDAS uses existing hardware or non-university participation to power the grid. This means that no new hardware costs are incurred. Further, DDAS never becomes obsolete; as the university upgrades it infrastructure so is the supercomputer upgraded. DDAS can perform exhaustive analysis on relatively large data sets, finding the best models resulting in better conclusions being draw from the data. Finally, DDAS takes advantage of otherwise idle CPU time, which increases the utilization of existing resources. The computers used by DDAS, when not in use by students or faculty, serve to further the project's computation.

7.2 The Limitations

DDAS in its current form extends the reach of researchers wishing to perform statistical analyses in the form of linear regressions. However, given its potential power, it is limited. Its power is tied to the number of computers in its grid. Currently there are approximately 64 university-owned computers in the grid. Even if the grid had 1024 computers, the number of additional columns in a potential data set may only grow by three or four for the execution time to remain constant.

As noted earlier, the dimensionality of data set that DDAS can handle is somewhat limited. The method to compute all possible regressions used by SAS appears to be far more efficient than the purely iterative method of DDAS, and so perhaps a rethinking of its method is in order. This potential change would allow DDAS to more fully realize its potential. It should be noted that

SAS' approach is not fully understood by anyone associated with DDAS and so may not be suitable or even correct with respect to the goals of the project.

Another limiting factor is that DDAS does not currently offer the ability to dynamically create and append non-linear and/or surface variable mutations to the original data set. These additional columns would allow DDAS to consider data sets in other ways than in terms of strictly linear models. It has been observed that there is benefit to such manipulation; however, this feature remains unavailable to researchers via DDAS.

A smaller limitation is that DDAS can only perform regressions with a single dependent variable. Any data set with two or more dependent variables may not take advantage of DDAS.

Finally, different types of files types should be considered than just comma separated value files. An XML format, complete with a validating schema could be created along with any other typical organization methods. This would save researchers the trouble of having to manipulate their data into a single format.

7.3 Future Work

- The creation of a web page that displayed project data. It would be visible to the public and could contain things like:
    - The purpose of the project
    - Researcher tutorials
    - The number of associated researchers
    - The number of jobs currently submitted
    - The number of jobs with active work units
    - Total computers working on the project
    - Total FLOPS per day
    - Total FLOPS since project inception
    - Total hours of CPU time per day

    - Total hours of CPU since project inception
- A more sophisticated problem management interface and backend should be created in the next iteration of DDAS. Researchers should be able to choose how their problems are controlled. For instance, if a researcher drops off more than one job, the researcher should be able to specify whether the jobs are analyzed serially or in parallel. Researchers should be able to cancel and/or pause their jobs.
- A confirmation when the researcher submits a job of that job's estimated time to completion based on the dimensions of the data set and the number of known, active clients attached to the project. This would prevent unwittingly submitting too large a data set or conversely too small a set.
- Multiple dependent variables could be added to the system. This would allow greater flexibility in terms of the kinds of data sets that researchers could submit to DDAS.
- Server-side data manipulation could be added. This would allow researchers to specify that their data be considered in other ways than strictly linear. To clarify, a set of columns may be subjected to a mathematical function like $log(x)$ or x^2. The resulting columns would be appended to the original data set. Also, surface variables could be added as part of this addition.
- The code base, particularly the server components would benefit from being re-factored. While a first pass at refactoring has been effected, enough new features were added that the benefits of the initial refactoring have been muted.
- A researcher should be able to log in and see the best results so far. Currently the researcher has to wait until the job is completed before seeing any results.
- Currently researchers are notified every 24 hours about the status of their jobs. The time period and participation in receiving e-mails should be researcher specifiable. Also, researchers receive one e-mail for each job. This should probably be changed to one comprehensive e-mail for all jobs belonging to the researcher in question.
- One could create a hybrid between all possible regressions and other heuristic methods like stepwise and backwards elimination. This would give researchers the ability to drop

off data sets with thousands of columns while still maintaining some of the benefits of all possible regressions.

- Currently, many components are not designed to gracefully recover from errors. Code should be rewritten where necessary to take advantage of C++ exceptions where appropriate.
- A web-service interface could be written so that researchers might send and receive information in other means that the HTML web-interface.
- The science application should check the dimensions of the data set against the scoping file before beginning *any* computation. If the dimensions do not agree, log the error and exit.
- Feed erroneous results to the validator and verify that they are discarded or at least not flagged as correct.
- An empirical or mathematical justification for the validator tolerance should be established. The current tolerance has no such justification.

8 Bibliography

1. **Top500.org.** Top 500 List - November 2007 (1-100). [Online] November 2007. [Cited: 3 30, 2008.] http://www.top500.org/list/2007/11/100.

2. **Top500.org.** Top 500 List - November 2007 (401-500). *Top 500 List.* [Online] 11 2007. [Cited: 3 30, 2008.] http://www.top500.org/list/2007/11/500.

3. **Wikipedia.org.** FLOPS. *Wikipedia.org.* [Online] 2008. [Cited: 3 30, 2008.] http://en.wikipedia.org/wiki/Teraflops#Records.

4. **The Unofficial BOINC Wikipedia.** BOINC FAQ: Introduction to BOINC - Unofficial BOINC Wiki. *The Unofficial BOINC Wikipedia.* [Online] 8 21, 2007. http://www.boinc-wiki.info/BOINC_FAQ:_Introduction_To_BOINC.

5. **Anderson, Dr. David.** YouTube - BOINC - Berkeley Open Infrastructure for Network Computing. *YouTube.com.* [Online] 8 26, 2006. http://www.youtube.com/watch?v=8iSRLIK-x6A.

6. **The Unofficial BOINC Wikipedia.** What Applications are Suitable for BOINC? *The Unofficial BOINC Wikipedia*. [Online] [Cited: 3 30, 2008.] http://www.boinc-wiki.info/What_Applications_are_Suitable_for_BOINC%3F.

7. **Statistics.com.** Backward Elimination. *Statistics.com.* [Online] June 18, 2008. [Cited: June 18, 2008.] http://en.wikipedia.org/wiki/Backward_elimination.

8. **Wikipedia.org.** Stepwise regression. *Wikipedia.org.* [Online] June 18, 2008. [Cited: June 18, 2008.] http://en.wikipedia.org/wiki/Stepwise_regression.

9. **Wikipedia, The Unofficial BOINC.** Canonical Result. *The Unofficial BOINC Wikipedia.* [Online] June 18, 2008. [Cited: June 18, 2008.] http://www.boinc-wiki.info/Canonical_Result.

10. —. Feeder Daemon. *The Unofficial BOINC Wikipedia.* [Online] June 18, 2008. [Cited: June 18, 2008.] http://www.boinc-wiki.info/Feeder_Daemon.

11. Quorum of Results. *The Unofficial BOINC Wikipedia.* [Online] June 18, 2008. [Cited: June 18, 2008.] http://www.boinc-wiki.info/Quorum_of_Results.

12. Scheduling Server. *The Unofficial BOINC Wikipedia.* [Online] 2 2, 2006. http://www.boinc-wiki.info/Scheduling_Server.

13. **Wikipedia.org.** Explained sum of squares. *Wikipedia.org.* [Online] June 18, 2008. [Cited: June 18, 2008.] http://en.wikipedia.org/wiki/Explained_sum_of_squares.

14. **BOINC Wikipedia.** BOINC Wikipedia. *SoftwarePrereqsUnix - BOINC - Trac.* [Online] http://boinc.berkeley.edu/trac/wiki/SoftwarePrereqsUnix.

15. **Foster.** Globus Toolkit 4.0: Key Concepts. *Globus Toolkit.* [Online] 5 8, 2005. www.globus.org/toolkit/docs/4.0/key/GT4_Primer_0.6.pdf .

16. **Lenstra, Dr. Arjen K.** Long Integer Package - LIP. *Homepage of Arjen.* [Online] http://www.win.tue.nl/~klenstra/.

17. **Unofficial BOINC Wikipedia.** BOINC Server-Side Daemon Program. *The Unofficial BOINC Wikipedia.* [Online] [Cited: 3 30, 2008.] http://www.boinc-wiki.info/BOINC_Server-Side_Daemon_Program.

18. Data Server - Unoffical BOINC Wiki. *The Unofficial BOINC Wikipedia.* [Online] 2 2, 2006. http://www.boinc-wiki.info/Data_Server.

19. BoincIntro - BOINC - Trac. *BOINC Wikipedia.* [Online] http://boinc.berkeley.edu/trac/wiki/BoincIntro.

20. **The Official BOINC Wikipedia.** CommIntro - BOINC - Trac. *The Official BOINC Wikipedia.* [Online] 2007. [Cited: 3 30, 2008.] http://boinc.berkeley.edu/trac/wiki/CommIntro.

9 Appendix

9.1 Definitions

BOINC: Berkeley Open Infrastructure for Network Computing – a generic framework for creating massive desktop grid supercomputers.

Canonical Result: The simplest and best result that is chosen from the set of results that are part of the quorum of results and is saved as the example result in the science database. (9)

Data File: The science application's data input file. It contains a subset of the problem's data set on which the computation is to be performed. It is specified in both the work unit and the results, that the server may know which files to send to the client for execution.

Definition Number: A unique model identifier used by DDAS. This number is the decimal value of the binary representation of the model derived by setting all considered columns to one and the remainder to zero.

Feeder: This is a BOINC server-side daemon program that fills up the ready- to-send queue with work units ready to be sent. The scheduler itself is too busy handling transaction with clients to do this task so the feeder daemon does this job. (10)

Quorum of Results: A collection (or set) of results that are ready for validation. The size of the set is a project specific value that determines how many successfully processed results have to be returned before there is an attempt to validate the work unit and credit can be allocated to those participants with results that were determined to be valid.

Result: A result is an instance of a work unit. A work unit is a template from which results can be created. Once the results are created they are stored in the project's database. Clients who request work receive a result, which contains reference to the science application and its input and output files, among other things. The result has various states, such as "ready", "complete",

or “computation error”. The project server, via the transition daemon uses these states to move a job from inception to completion. (11)

Scheduling Server: This server coordinates the work that is issued to make the best use of the computers available to process the work that is ready for issue. (12)

Science Application: A science application is the project specific executable which performs the scientific computation on the client computer. Every BOINC project, by definition, has at least one science application. When you join a project, the BOINC manager downloads this application, along with an appropriate data file to perform the client-side computation. The data that results from this processing is then returned to the project and new data is downloaded. If a project updates its science application, the BOINC client software will automatically recognize and download the new version as part of its automatic update feature.

Scoping File: The scoping file delineates the start, end and dimensions of the data set. It is DDAS specific. Each work unit has a unique scoping file, telling the science application how to initialize properly.

SSE: In statistics, an explained sum of squares (ESS) is the sum of squared predicted values in a standard regression model (for example $y_i = a + bx_i + \varepsilon_i$), where y_i is the response variable, x_i is the explanatory variable, a and b are coefficients, i indexes the observations from 1 to n, and ε_i is the error term. In general, the less the ESS, the better the model performs in its estimation. (13)

State File: The state file is a long term storage facilitator. Each work unit is assigned a state file to which it may write its intermediate results. The purpose is to preserve work even in the event of unforeseen interruption.

Task: See Result.

Univariate Scaling: For the purposes of this paper, a univariate scaling would mean taking a single variable, x, and scaling it exponentially x^2, x^3… x^n and using those scalings as inputs for the regression computations.

Work Unit: The result of dividing the large data set into smaller, more tractable pieces is a number of work units. The work unit contains, among other things, the scope of the work to be performed and a reference to the files required by the client to perform the work. A work unit defines the work to be performed by client; it is a template from which results can be created. Results are instances of work units – results are what are sent to a client, not the work unit itself. A work unit has one or more results which are distributed to clients.

9.2 Use Cases

Researcher Login

Actor(s):

Researcher

Pre-condition(s):

The researcher has contacted the server administrator and received a researcher account, along with username and password, for the DDAS project.

Post-condition(s):

The researcher is granted access to the DDAS web interface.

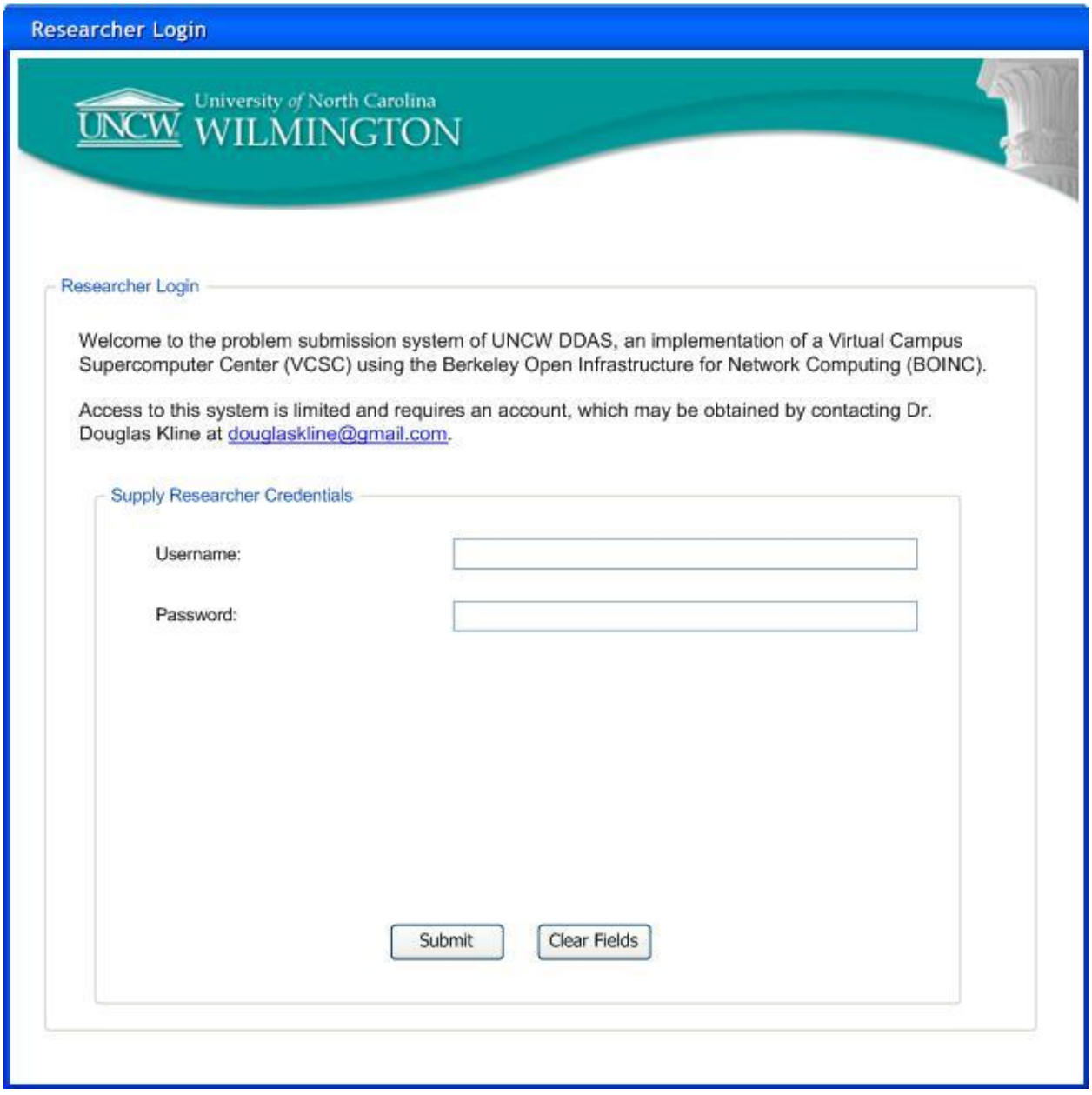
Happy Path:

The researcher uses correct username and password and is granted access to the system for some duration.

Alternative Path(s):

The researcher uses incorrect name and/or password and is not granted access to the system. The researcher must attempt to log in again or contact the server administrator to reset his or her username and/or password.

Researcher Login Mockup:



Researcher Submits Work

Description:

The researcher has a data set on which he or she would like a statistical analysis performed. The researcher logs into the system and uploads the data set.

Actor(s):

Researcher

Pre-condition(s):

The researcher is logged into the system.

Post-condition(s):

The researcher has successfully uploaded his data. The server has a copy of the data set and is primed for work generation.

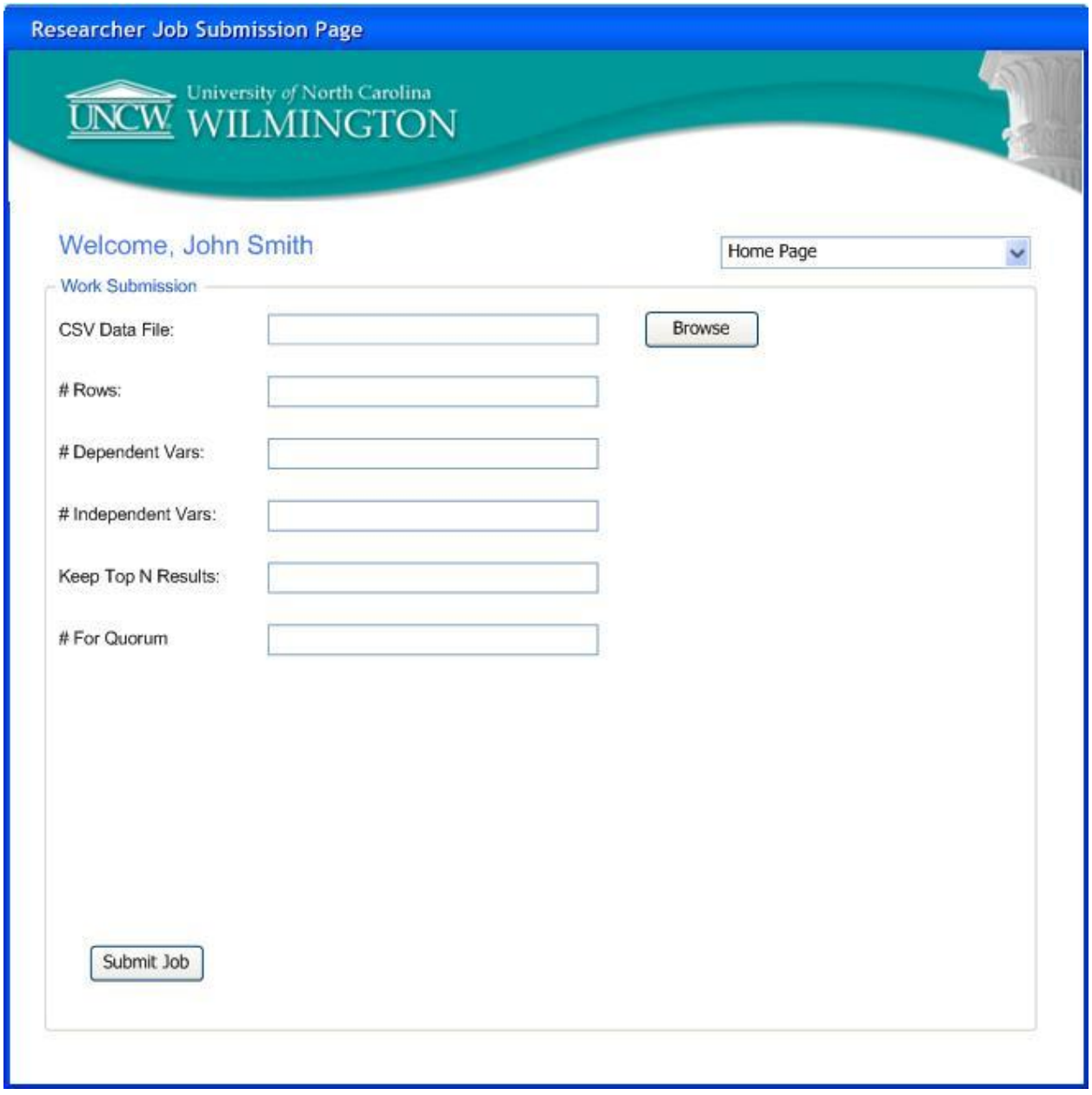
Happy Path:

The researcher successfully uploads his or her data set to the DDAS server, and receives an acknowledgement in the form of a web page.

Alternative Path(s):

The upload fails and the researcher must attempt to upload his or her data again. The researcher will receive a failure to upload web page if this occurs. If the upload fails continuously, the researcher must have the server administrator manually add the data set to the server.

Researcher Submits Work Mockup:



Researcher Views Work Status

Description:

The researcher checks in periodically to see the status of a job or jobs he or she has submitted.

Actor(s):

Researcher

Pre-condition(s):

The researcher must be logged in to perform this function.

Post-condition(s):

The researcher receives a page containing summaries of the jobs he or she has submitted previously.

Happy Path:

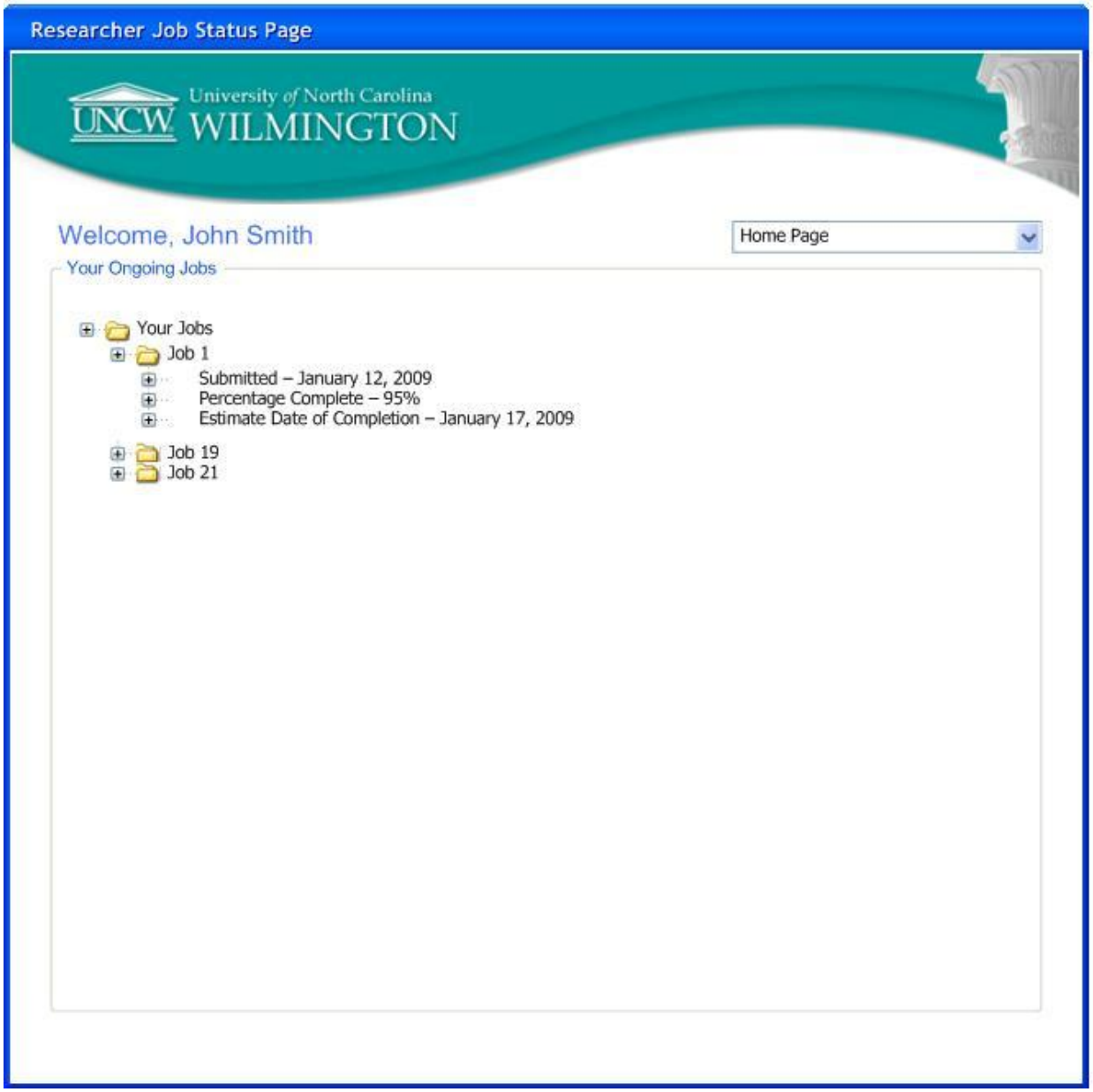
The researcher requests a summary of his or her jobs. There are jobs to summarize and they are transmitted in the form of HTML to the researcher's web browser.

Alternative Path(s):

There are no jobs to summarize and the researcher receives a web page confirming this.

There are jobs to summarize but the researcher receives indication that there are not. The researcher must contact the project administrator to find the cause of the problem.

Researcher Views Job Status Mockup:



Researcher Downloads Job Result(s)

Description:

The researcher downloads the output file from the statistical analysis from the DDAS web interface.

Actor(s):

Researcher

Pre-condition(s):

The researcher must be logged in.

The researcher must have completed jobs.

Post-condition(s):

The researcher has a local copy of the result file, downloaded from the DDAS server.

Happy Path:

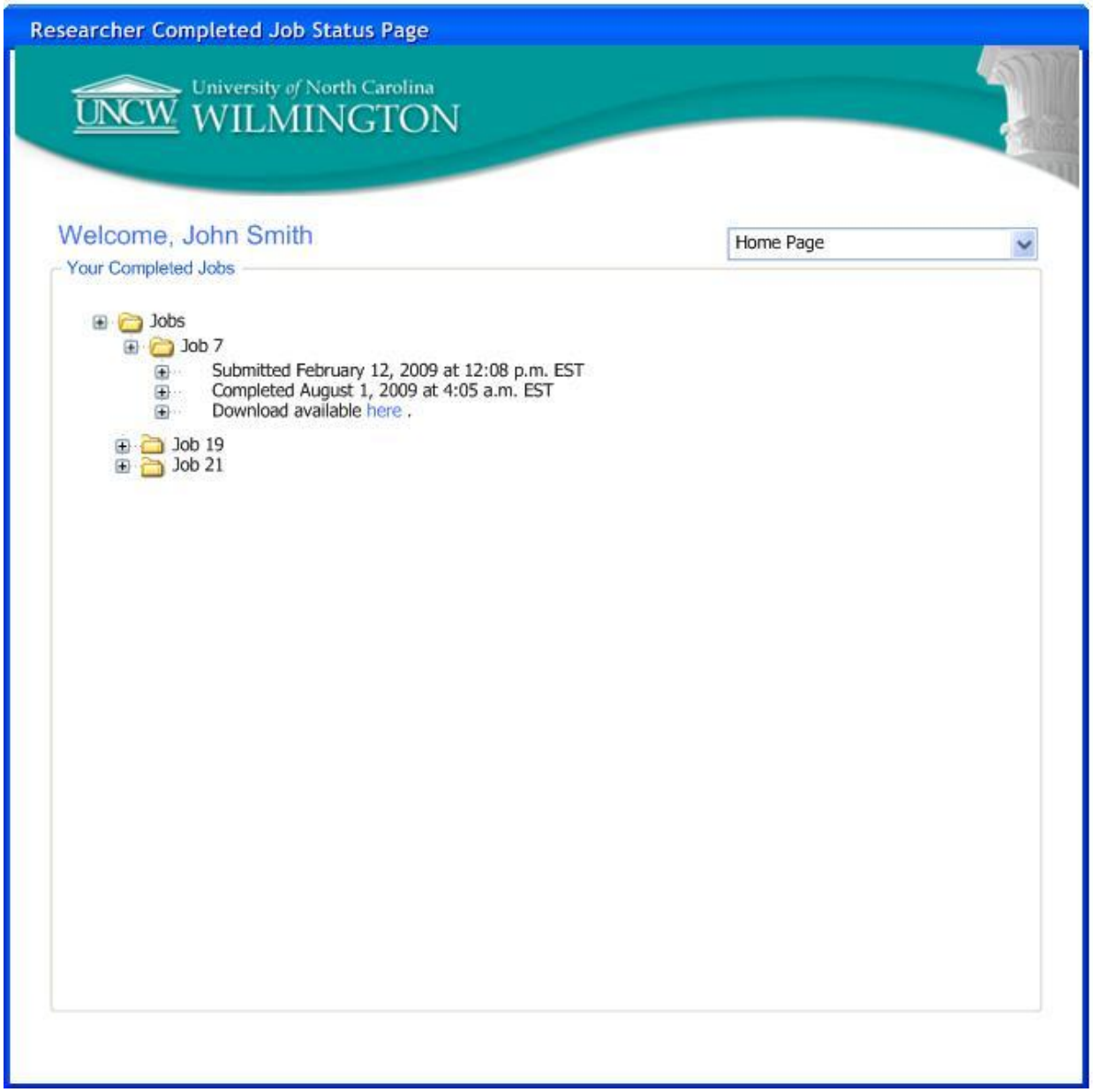
The researcher is logged in and has one or more completed jobs. The researcher chooses the view completed jobs page from his or her web interface. The researcher chooses to download the result file from the web server.

Alternative Path(s):

The researcher has no completed jobs and is therefore unable to download the related result files. The researcher must wait until at least one of his or her tasks are complete before downloading the result file.

The researcher has completed jobs but is unable to view them. In this case the researcher must contact the server administrator. Depending on the duration since job completion, the server administrator may be able to provide a copy of the result file.

Researcher Download Results Mockup:



9.3 DDAS External Dependencies

1. BOINC – Berkeley Open Infrastructure for Network Computing:
    a. http://boinc.berkeley.edu/
    b. BOINC is the de facto standard for desktop Grid computing.
2. GNU Scientific Library:
    a. http://www.gnu.org/software/gsl/
    b. The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It is free software under the GNU General Public License. The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite.
3. Visual Studio 2005:
    a. http://msdn2.microsoft.com/en-us/vstudio/default.aspx
    b. Microsoft's Multilingual IDE.
4. Red Hat Enterprise Linux:
    a. http://www.redhat.com/rhel/?s_kwcid=redhat%20linux|1081334061
    b. The world's leading open source application platform
5. Long Integer Package:
    a. http://www.win.tue.nl/~klenstra/
    b. Dr. Lenstra's C package for arbitrary length integer arithmetic
6. Subversion
    a. http://subversion.tigris.org/
    b. Source control package
7. Google Code
    a. http://code.google.com
    b. Google's free code repository, download repository and Wikipedia. DDAS' source code may be found at: http://code.google.com/p/uncwddas/.
8. Xerces XML Library

a. http://xerces.apache.org/xerces-c/

b. Xerces C++ XML Parser

9.4 Tables

9.4.1 Table 1 BOINC Software Prerequisites

Software Packages	Server	Core Client	BOINC Mgr
Subversion (SVN) is needed to obtain the BOINC source code	X	X	X
GNU tools make 3.79+, m4 1.4+, libtool 1.4+, autoconf 2.58+, automake 1.8+, GCC 3.0.4+	X	X	X
pkg-config 0.15+	X	X	X
Python 2.2+ with MySQLdb module 0.9.2+ and xml module	X		
MySQL 4.0.9+ (with mysql-dev(el), and mysql-client)	X		
Apache w/ mod_ssl and PHP5+	X		
PHP5 w/ cli support and GD (packages php5-cli and php5-gd)	X		
OpenSSL version 0.9.8+	X	X	
libcurl version 7.15.5+		X	

wxWidgets 2.8.3	X
Graphics libraries **GL, GLU, GLUT, or freeglut (static versions only)**	
jpeglib, X11 libraries and include files	X

(14)

9.4.2 Table 2 DDAS vs. Globus Toolkit

Quality	BOINC	Globus Toolkit
Architecture	Client-Server	Service Oriented
Expected Data Throughput	Low to Medium	Varied
Expected Data/Computation Ratio	Low to Medium	Varied
Typical Node Trust Level	Very Low	Moderate
Typical Project Size	Global	(Multi-)Organizational
Communication Mechanisms	XML, HTTP	Web-Services Protocols
Heterogeneity of Problem	Narrow or Singular	Varied
Supported Languages	C, C++, Fortran	Java, C, Python
Security	RSA, MD5	RSA , TLS, WS-Security, WS-SecureConversation
Computational Redundancy	Specifically Addressed	?
Inter-node Communication	No	Yes
Node Addressing Method	BOINC Manager	WS-Addressing,
Resource Addressing Method	BOINC Manager	WS Resource Framework
Node Notification Method	BOINC Manager	WS-Notification
Data Transport Protocols	HTTP, FTP	GridFTP (FTP extension)
Interactional Complexity	Low	High

Typically Volunteer Driven	Yes	No
Typical Node Geo-Dispersion	Global	(Multi-)Organizational
Typical Data Geo-Dispersion	Centralized	Varied
Platform Independence	Medium	High
Run User Submitted Data	Yes	Yes
Run User Submitted Programs	No	Yes
Standards Adherence	Low	High (SOAP, WSDL, XML)

(15)

9.4.3 Table 3 DDAS vs. SAS (Basic Iris Data Set)

Basic Iris Data Set	DDAS	SAS
SSE	**6.957763**	**6.95776**
Y Intercept	**-0.807916**	**-0.80792**
X1	**-0.109741**	**-0.10974**
X2	**-0.044240**	**-0.04424**
X3	**0.227001**	**0.22700**
X4	**0.609894**	**0.60989**

9.5 Project Timeline

- **Spring 2007** (MIS 592)**:**
  - BOINC server configured and working
  - Apache server configured and working
  - Validator daemon beta-version working
  - Assimilator daemon unimplemented
  - Work generator daemon unimplemented
  - Researcher web interface beta-version working
- **Spring 2008:**
  - Windows® client beta-version implemented
  - Validator daemon revised to version 1.0
  - Assimilator daemon beta-version implemented
  - Work generator daemon beta-version implemented
- **Summer 2008:**
  - Complete Windows® client
  - Implement client checkpointing
  - Revise Windows® screensaver
  - Revise Validator daemon
  - Revise Assimilator daemon
  - Revise Work Generator daemon
  - Revise researcher web interface to allow greater control over variables
  - Document code

9.6 DDAS Final Deliverables

- **Server Deliverables**
  1. DDAS Validator Daemon
  2. DDAS Assimilator Daemon
  3. DDAS Work Generator Daemon
  4. DDAS Science Database
  5. Researcher Web Interface
  6. Documentation
  7. Code API
- **Client Deliverables**
  1. Windows® APR Client Executable
  2. Windows® Screensaver
  3. Documentation
  4. Code API

9.7 Potential Research Questions

- All possible regressions compared with other, quicker, regression methodologies: what is the benefit of doing all possible regressions? How frequently do all possible regressions return a better result? At what cost? Is this cost worth it?
- What is the upper bound on problem size with respect to computational feasibility for DDAS? Given that the work unit count increases exponentially with respect to the number of independent variables, what is the maximum feasible number of independent variables?
- What is the optimal work unit size? Given a data set of a certain size, a network connection and a computer of certain speed, what is the optimal work unit size? How can this be dynamically created?
- Can using matrices of full rank as is done by SAS provide non-heuristic results more quickly than a pure brute force approach? If so, how can this approach be integrated into DDAS and how does this affect the size of data set it may consider?

9.8 Acknowledgements

- Dr. Douglas Kline
- Steve Sutton
- Royce Nobles
- Shaun Border

- University of North Carolina, Wilmington
- University of California, Berkeley

- The BOINC Contributors
- The Official BOINC Wikipedia
- The Unofficial BOINC Wikipedia
- Red Bull™ Energy Drink

9.9 Example Science Application Output File

```
<DDAS_result_set>

  <data_row SSE="121.554031" binary="11011110101101010"
definition_number="48490">
    <regression_coefficient id="0" value="-0.853902"/>
    <regression_coefficient id="1" value="0.139655"/>
    <regression_coefficient id="2" value="0.000000"/>
    <regression_coefficient id="3" value="60700454349.505081"/>
    <regression_coefficient id="4" value="122173470.563664"/>
    <regression_coefficient id="5" value="0.005888"/>
    <regression_coefficient id="6" value="0.002641"/>
    <regression_coefficient id="7" value="0.000000"/>
    <regression_coefficient id="8" value="0.013296"/>
    <regression_coefficient id="9" value="0.000000"/>
    <regression_coefficient id="10" value="-60700454349.505196"/>
    <regression_coefficient id="11" value="0.000147"/>
    <regression_coefficient id="12" value="0.000000"/>
    <regression_coefficient id="13" value="-122173470.565982"/>
    <regression_coefficient id="14" value="0.000000"/>
    <regression_coefficient id="15" value="0.020480"/>
    <regression_coefficient id="16" value="0.000000"/>
  </data_row>

  <data_row SSE="121.527382" binary="11110111111001100"
definition_number="61388">
    <regression_coefficient id="0" value="-0.846253"/>
    <regression_coefficient id="1" value="317898913.499358"/>
    <regression_coefficient id="2" value="-70892017866.340561"/>
    <regression_coefficient id="3" value="-64380844.716590"/>
    <regression_coefficient id="4" value="0.000000"/>
    <regression_coefficient id="5" value="-821488951067.563840"/>
    <regression_coefficient id="6" value="0.002661"/>
    <regression_coefficient id="7" value="0.144446"/>
    <regression_coefficient id="8" value="-317898913.486398"/>
    <regression_coefficient id="9" value="70892017866.340424"/>
    <regression_coefficient id="10" value="-0.002303"/>
    <regression_coefficient id="11" value="0.000000"/>
    <regression_coefficient id="12" value="0.000000"/>
    <regression_coefficient id="13" value="64380844.722433"/>
    <regression_coefficient id="14" value="821488951067.585210"/>
    <regression_coefficient id="15" value="0.000000"/>
    <regression_coefficient id="16" value="0.000000"/>
  </data_row>

</DDAS_result_set>
```