The InfoServer – An Information Server for NetSolve Users

Jakob Østergaard (jakob@unthought.net)
Innovative Computing Laboratory*

October 12, 2000

A problem in the NetSolve system today is, that users and user applications are unable to efficiently and easily retrieve information about running jobs, server status, job history etc. This paper suggests the integration of an *Information Server* into the NetSolve system. The Information Server is essentially a very simple database that receive information from the NetSolve Agent, and serve out that information as requested by remote users.

1 Status

The InfoServer exists today in a prototype implementation. It is roughly 2K lines of ANSI C, where 2/3 of the code is socket handling and other necessary system interaction. The rest of the code is a very simple database engine – which will have to be rewritten in order to suit the needs outlined in this paper.

This paper is submitted to the people involved with the NetSolve project, and it is a *request for comments* on the basic ideas described herein.

2 Problems

There is a number of problems with getting relevant information out from the NetSolve system today – as well as problems with information management in general. Today the only means of getting out current and historical server and job information from the system, is to read the NetSolve Agent log file. A list of some of the problems with that approach are:

*ICL, University of Tennessee, Knoxville, and Oak Ridge National Laboratory

1. The Agent log is not thorough, for example time-stamps are found missing some places, and it is not always possible to decide which messages are related to each other (esp. in job submission/tracking)
2. User programs such as the Monitoring GUI have no easy way of actually getting to the Agent log.
3. Reading the log is inefficient. Imagine 10 user programs retrieving a 10 MB log file every second to keep track of changes. This will require a one Gigabit network connection to keep 10 users informed about the status of the NetSolve system. And even then, they will have at least a one second latency on the information, unless they poll more often.
4. "Rotating" the log, eg. renaming the old log and starting a new one (or just somehow shortening the existing log), in order to limit the amount of data consumed by the Agent Log, is bound to cause problems for users trying to deduce for example what servers are registered in the system. What if the log-lines containing the server registration are cut off, and the servers are not re-started ? Then the user reading the log has no way of telling the servers exist, or what problems they may provide solvers for.

The InfoServer seeks to eliminate these problems by providing an "Information Service", a networked simple database-like service tailored to fit our needs. Following is a list of arguments for how the InfoServer will solve the problems highlighted by the above list.

1. By providing a very simple "insertion" function, the Agent can submit and classify information with a simple call that enforces or encourages thoroughness. Instead of simply writing a text-string of some arbitrary format, the call will require classification of the information with keys and values. See Section 3 for an idea of the internal representation of these data.
2. As the InfoServer is a networked service, a client will simply connect to the InfoServer and submit it's requests for data. See Section 6 for the implementation details, and Section 5 for a discussion of the security concerns.
3. Providing actual *queries* that can filter out irrelevant information, a user program will be able to retrieve information as needed. This will allow many clients to follow the information available from the Agent without saturating the network. A query *could* be implemented to even let the user programs avoid polling entirely – this would further decrease network bandwidth consumption and significantly decrease the latency in the propagation of information. This last type of request is not further discussed in this paper.
4. Management of information can be done in clever ways, when we control how data is both written, kept, and read. For example, it should be

simple to limit the number of records kept in the database, limiting the number on a per-record-type basis instead of on an all-record basis (as with ordinary log-file rotation). This issue is also slightly touched upon in Section 6.

3 Data representation

There are multiple goals to try to satisfy in the InfoServer. Our main concerns with the data representation is, that it must provide:

- Simple queries, including the ability to actually distinguish server names from problem names etc. to make it possible to do queries on these "fields"
- Efficient retrieval of data – one server must be able to easily serve tens or hundreds of user programs
- Very simple insertion mechanism – for minimal changes in the NetSolve Agent

One "record" in the database will consist of a variable number of "fields". Each field has a "key" and a "value". An example of this is shown below:

$\text{key}_a = \text{value}_1$	$\text{key}_b = \text{value}_2$	$\text{key}_c = \text{value}_3$

We can allow queries including one or more keys (as we see fit), while keeping the InfoServer very simple still. The InfoServer will be given records such as

`TYPE` = "JOB_COMPLETED"	`HOSTNAME` = "torc0"	`JOB_ID` = "42"

without ever knowing what a hostname or a job_id is. These are just keys, and keys have values – the InfoServer knows no more, and doesn't need to know more.

4 Queries

The queries we wish to be able to perform, should be sufficiently covered by the following query language (in EBNF[1]):

```
     space = ' ', {' '};
       and = 'and';
        or = 'or';
  equality = '=';
  succeeds = '>';
     alnum = (* letters a-z and A-Z as well as digits 0-9 *)
comparison = equality | succeeds;
```

[1]ISO/IEC 14977

```
combination = and | or;
      query = 'get', space, qexp, ';';
       qexp = keyval
            | keyreq
            | '(', [space], qexp, [space], ')'
            | qexp, space, combination, space, qexp;
     keyreq = keyname;
     keyval = keyname, [space], comparison, [space], value;
    keyname = alnum, {alnum};
      value = '"', {alnum}, '"';
```

Some examples of valid queries on a fictional database are given here:

```
get␣(TYPE="JOB_COMPLETED"␣or␣TYPE="JOB_FAILED")␣and␣JOB_ID;
get␣␣HOSTNAME␣=␣"torc0"␣and␣WORKLOAD;
get␣WORKLOAD␣;
```

The queries will return

1. All events of type `JOB_COMPLETED` or `JOB_FAILED` also containing a `JOB_ID`
2. All events containing a `WORKLOAD` key where the `HOSTNAME` key has the value **torc0**
3. All events containing a `WORKLOAD` key

The actual query engine to implement this is not yet written. It should however be simple to build both the parser for the query language, as well as a simple query engine that has *reasonable* run-time performance for *typical* queries. We are not developing a state-of-the-art database system here, the InfoServer should provide good performance for common queries, no more and no less.

5 Security

There are several aspects of security when dealing with an information service such as the one presented here. Some aspects are of no concern now but could become concerns later on, some are always relevant to consider. In any case, the InfoServer should, as far as possible, make it simple to introduce access controls etc. later on if or when it is deemed necessary or desirable.

Since the InfoServer will do no more than receive, store and publish information, security should be well considered at an early stage. Even though our requirements for security are low (just about negligible) today, the clever thing is to consider security today while the system is small.

5.1 Exploits and Denial of Service

By keeping the InfoServer self-contained and simple, and using safe coding practices (not using buffers on the stack, not using unsafe library calls, and not relying on any external programs for anything), it should be possible to keep the system free from buffer overflow exploits and other common security holes.

An important point is that the InfoServer does not require any special privileges – it can be run as any user on a system, if only that user has normal disk access. A fairly safe installation can be done by making sure that the InfoServer is running as a special user that has minimal rights.

Denial of Service (DoS) attacks cannot be completely guarded against. There is no way to assure that only "friendly" user programs can connect. Currently there is no DoS prevention code in the InfoServer, but some counter measures could be built in easily, as the request and connection handling code is neatly isolated in one place.

5.2 Basic Infrastructure

In order to allow for later expansion of the security system within the InfoServer, there are some basic needs we must satisfy in the system. This is best done from the beginning of the lifetime of the system, while it is still simple.

However, because of the generality of the representation of information, there may not be a lot of implementation to do at all. For example, all records containing host-based information will contain a `HOSTNAME` key and value, and the access control mechanism (Section 5.4) can use that information to allow or disallow access to that particular record.

Comments and ideas on the problem of classification of information provided by and stored within the InfoServer will be greatly appreciated.

5.3 Authentication

There is currently no authentication of information sources (the NetSolve Agent(s)) and consumers (the user programs). We choose to believe that whoever (presumably the Agent) submits log information is someone who is telling the truth.

As the NetSolve system today has no means of user authentication what so ever, there is also no user authentication in the InfoServer. However, if user authentication is one day added to NetSolve, it should be simple to introduce this into the InfoServer as well. A good security infrastructure in the InfoServer should make this reality.

5.4 Access control

A reference monitor[2] should be implemented, and it should be ensured that the reference monitor is invoked whenever information is retrieved from the InfoServer.

Later on, when an actual security policy is decided upon, it will be trivial to extend the reference monitor to enforce the policy decided upon. A "dummy" reference monitor should be implemented immediately, this one could just allow anyone to do anything, implementing no real security at all. Having the infrastructure in place is crucial for later implementation of a security policy.

The access control system in the InfoServer should be easily expandable to provide discretionary access control on a per-record/per-user basis, once a user authentication mechanism is included within NetSolve. This goal seems immediately achievable given the infrastructure described here.

5.5 Other ideas

Other security mechanisms in common use in data processing systems include: Audit trails, labeled security and mandatory access control (MAC). Currently it seems as massive overkill to even consider implementing any of those, although they would all be relatively simple to include. These measures currently do not make sense to consider in the NetSolve environment.

6 Implementation details

An actual implementation of the InfoServer exists – this code is a proof of concept, but the code has been built with real world use in mind. The code should be stable and efficient, and whatever changes may be needed, we should be able to put them into the current code without too much hassle.

The InfoServer is a separate process that can either run on the Agent machine, or on a separate machine on the network. Initially it will be desirable to have the Agent spawn the InfoServer, to make it's presence as "transparent" as possible. Later on, perhaps for scalability reasons, it could become preferable to have the InfoServer running on a dedicated machine.

The current implementation of the InfoServer supports both modes of operation, the only distinction being whether the Agent data are received via. (networked) TCP or (local) UNIX sockets.

Multiple readers and writers are supported in either mode, as anyone can connect to the InfoServer and submit input data or data requests (queries). See Section 5 for the security discussion.

[2]See DoD 5200.28-STD *Department of Defense Trusted Computer System Evaluation Criteria*

The main focus in the current implementation has been to keep the InfoServer as self-contained as possible (requiring no extra libraries and interfering minimally with the existing NetSolve code), and as small and simple as possible (but no simpler).

6.1 Data management

Being the one entity that controls reception, storage and retrieval of information, allows the InfoServer to play clever tricks with the disposal of obsolete or "old" data. We do not have to rely on traditional "log rotation", we can actually limit the number of records kept in the database based on the content of the records.

For example, it will be relevant to keep a certain number of server registration records on a per-server basis, while it may be relevant to limit the number of workload reports on a per-time basis. Actually structuring the information as outlined in Section 3 will allow us to do this.