A general purpose data analysis monitoring system with case studies from the National Fusion Grid and the DIII-D MDSplus between pulse analysis system

S. Flanagan[1], J.R. Burruss[1], C. Ludescher[2], D.C. McCune[2], Q. Peng[1], L. Randerson[2], D.P. Schissel[1]

[1]General Atomics, P.O. Box 85608, San Diego, California 92186-5608

email: flanagan@fusion.gat.com, phone: (858) 455-4074

[2]Princeton University, Princeton, New Jersey 08543

As computing infrastructures become more complex, it is important to centralize information order to efficiently monitor and maintain computing processes. A general monitoring system has been developed at the DIII-D National Fusion Facility that combines code run status, data analysis tracking, logfile access, complex error detection, and expert system capabilities. The monitoring system's flexibility and ease of deployment have enabled it to be successfully applied to two significantly different computing environments.

At DIII-D, the system is being used as the Data Analysis Monitor (http://nssrv1.gat.com:8000/dam) to allow both application scientists and computer scientists to monitor the status of between–pulse MDSplus dispatched data analysis codes. The monitoring system is also being used by the National Fusion Collaboratory Project, as the Fusion Grid Monitor (http://nssrv1.gat.com:8000/fgm), to track multiple asynchronous complex code runs on the FusionGrid.

A GENERAL PUPOSE MONITOR FOR COMPLEX COMPUTING ENVIRONMENTS

- Centralizing information becomes important as computing environments and analysis processes become more complex
  - Clusters
  - Remote / Grid Computing
  - Between-Pulse / Automated Analysis Processes
  - Distributed Processes

- A general monitoring system has been designed to…
  - Centralize information for efficient maintenance
  - Assist in maintaining a manageable workload
  - Be applicable to significantly different computing environments
  - Offer simple end-user monitoring of the state of code runs
  - Detect discrepancies in diagnostic measurements and analysis results

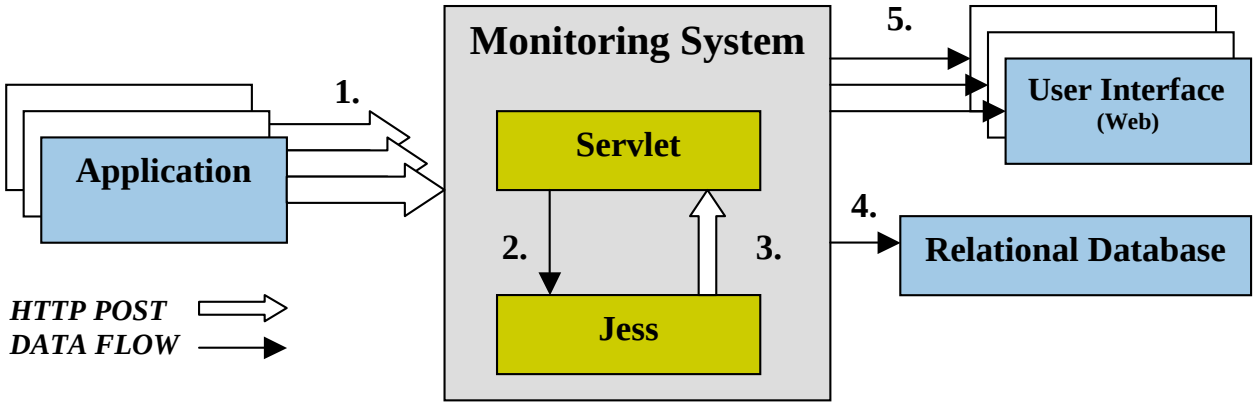
MONITORING WITHOUT WASTING RESOURCES

(CPU CYCLES, NETWORK TRAFFIC, ETC…)

- Why design a new monitoring system?
  - Many existing monitoring systems are undesirably CPU intensive
  - Some require constant polling of data
  - Others waste resources on intensive numerical computations
  - Or, include expert systems that wind up spending too much time evaluating logic

- How does the new monitoring system solve some of these issues?
  - It has been designed as a monitoring system that is not CPU intensive, does not rely on polling data, and does not (itself) handle numerically intensive calculations
  - Instead, it receives its information generically (via HTTP post requests) from remote computers
  - This allows any CPU intensive data analysis to occur across several different machines, allowing the monitor to receive, post, and evaluate only that information with which it has been provided
  - It also includes an expert system shell, which gives it intelligence to perform beyond the role of a simple posting mechanism

DATA FLOW THROUGH THE MONITORING SYSTEM



1. Applications make HTTP posts to the monitor's Java servlet
2. Information is passed to the Java Expert System Shell, Jess
3. Jess may make further posts to the servlet based on the incoming message, and the rules defined in CLIPS (C Language Integrated Production System)
4. All posts are logged to a relational database
5. Each connected user receives an updated HTML display through server push, or client pull

INFORMATION IS RECEIVED VIA AN HTTP POST REQUEST AND DISPLAYED THROUGH A WEB INTERFACE

- How does the monitor actually receive information?
  - Runs as a Java Servlet, using JSDK2.1
  - Receives information from HTTP post requests
  - Information can be posted from any host, however the monitor will only acknowledge messages from authorized/specified hosts

- How and Where does this information actually get posted?
  - All posts are displayed on the HTML web interface
  - HTML is dynamically created, and sent to the client when a new fact is declared
  - Server push, allows the monitor to update each client in real time, as information becomes available
    - Netscape, Mozilla, Opera…
  - Client pull is used with Internet Explorer, which does not properly support server push

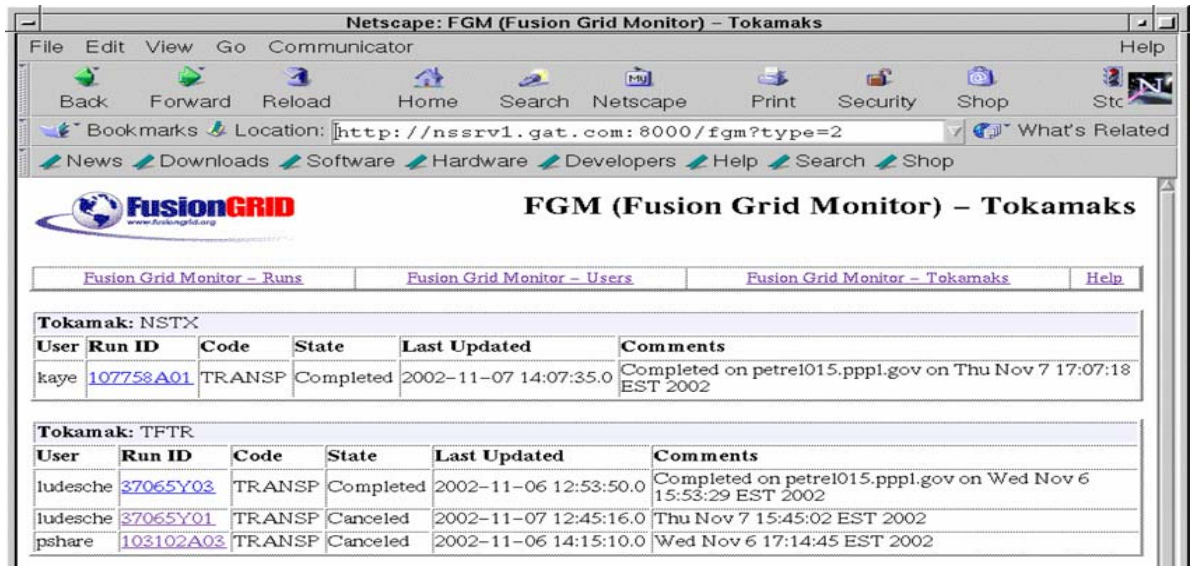
A RULE BASED EXPERT SYSTEM SHELL GIVES THE MONITOR THE CAPACITY TO “REASON”

- Uses Jess, the Java Expert System Shell
  - This allows the monitor to accept information, evaluate rules, and take action on knowledge it has obtained
  - Allows the monitor to perform beyond a simple posting mechanism, such as determining if a code has failed to report

- Any expert system is composed of a knowledge base, and a set of rules to which that knowledge is applied
  - A knowledge base is a set of facts known to the expert system
  - Facts from this knowledge base are applied to rules, which when satisfied, define results or actions to be taken
  - This could include firing new analysis codes, or updating the knowledge base

- The monitor can use Jess for expanding and customizing error tracking
  - The monitor can "decide" when codes have failed to report
  - Jess can also be used to execute certain actions upon errors

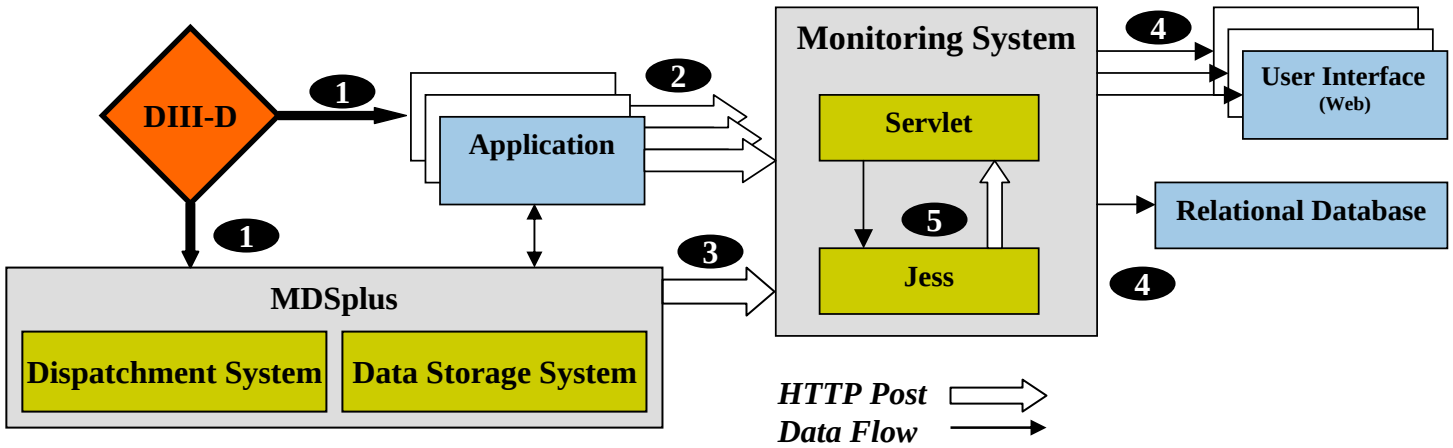
THE USER INTERFACE FOR THE MONITOR IS PROVIDED AS A WEB PAGE (VIA HTML)



- Users track and monitor the state of applications on FusionGrid
  - Output dynamically via HTML
    - Server Push, and Client Pull
  - Display can be sorted by…
    - Run ID, User, or Tokamak
  - Detailed log run files accessible
    - rcp, or anonymous FTP
- Code maintenance notification
  - Users are notified of…
    - queuing turned off, code rebuilt, queue restarted, etc…

AT DIII-D, DAM MONITORS MDSPLUS AND THE BETWEEN PULSE ANALYSIS SOFTWARE

- DAM is the between pulse monitoring system being used at DIII-D
  1. During operations, raw data is obtained and several data analysis programs are executed (about 25 for DIII-D)
  2. DAM receives information about the between pulse analysis software.
  3. Status messages from the MDSplus phases that dispatch these programs are also sent to DAM
  4. DAM will record each message into a database, and displays the new information to all users
  5. If an analysis program fails to report, DAM is capable of generating an error message



MONITORING AND TRACKING ANALYSIS CODES WITH DAM

- Tracking of dispatched routines:
  - When a pulse happens, several analysis routines are dispatched
  - Analyzed data is then stored in MDSplus
  - By inserting “sendpost” commands to perform HTTP posts to the monitor, DAM can easily track the status of MDSplus data loading, or any other dispatched analysis routines

- Examples of dispatched routines that are monitored:
  - At DIII-D, loaddata.pro launches several routines. By inserting a single sendpost command, DAM is provided with the final status of several routines
  - EFIT sends DAM messages that allow it to monitor the completion of calculations, and when EFIT data has been loaded into MDSplus
  - DAM also monitors a comparison between the experimentally measured neutron rate and the expected neutron emission, RDD0D

ERROR REPORTING FOR SPECIFIC PHYSICS ANALYSIS CODES:
(EXAMPLE: CALCULATED VS. MEASURED NEUTRON RATES)

- Physics Error Detection
  - DAM can be used to provide error detection in specific physics analysis codes
  - A comparison algorithm was written to compare the expected neutron rate, RDD0D, to the measured neutron emission, PLASTIC
  - This routine posts several information, including whether the data was a sufficient match, or whether the data was even available

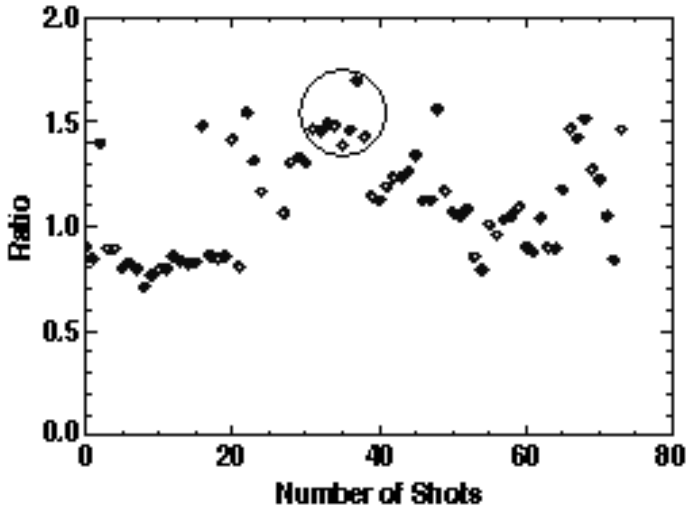
- The comparison algorithm for RDD0D
  - Finds the largest time interval (over 100ms) of steady state beams.
  - Measured data is averaged over a 10ms period in order to place it on the same time base as RDD0D
  - Comparison of the RDD0D and PLASTIC is then done within the time interval, to produce a ratio at each time point
  - From the ratios, it is possible to determine the systematic error for the current pulse.
  - The calculated neutron rate should be within 30% of the measured rate

STATISTICS OBTAINED BY COMPARING THE MEASURED NEUTRON RATE AND EXPECTED NEUTRON EMMISION, RDD0D

- At DIII-D, DAM monitors a comparison between the experimentally measured neutron rate and the expected neutron emission, RDD0D
  - This comparison algorithm produces a ratio that is used to determine the systematic error.
  - The calculated neutron rate should be within 30% of the measured rate

- Statistics were created for the month of June 2001
  - Roughly 18% of pulses (about 80) had an acceptable time window for comparison
  - 75% of which fell within acceptable error limit.
  - 25% of which fell outside acceptable error limit

  - On June 6th, all shots had high degree of error. Significant methane pumping was performed on this day



- These statistics were obtained based on error messages reported to DAM

THE NATIONAL FUSIONGRID

The Grid refers to the infrastructure that enables the integrated collaborative use of high–end computers, networks, databases, and scientific instruments owned and managed by multiple organizations.

- Resource owners decide which of their resources to contribute
  - Authentication keeps the Grid secure
- Not all Grid members can use all resources
  - Authorization allows finer grain access control to resources
- Single sign–on eliminates the need to log into multiple machines
  - User logs into the Grid only one time
  - Delegated credential accompanies all resource requests
- Much implementation, then, is hidden from users
  - It is important that grid members be able to easily monitor application runs and access logfiles

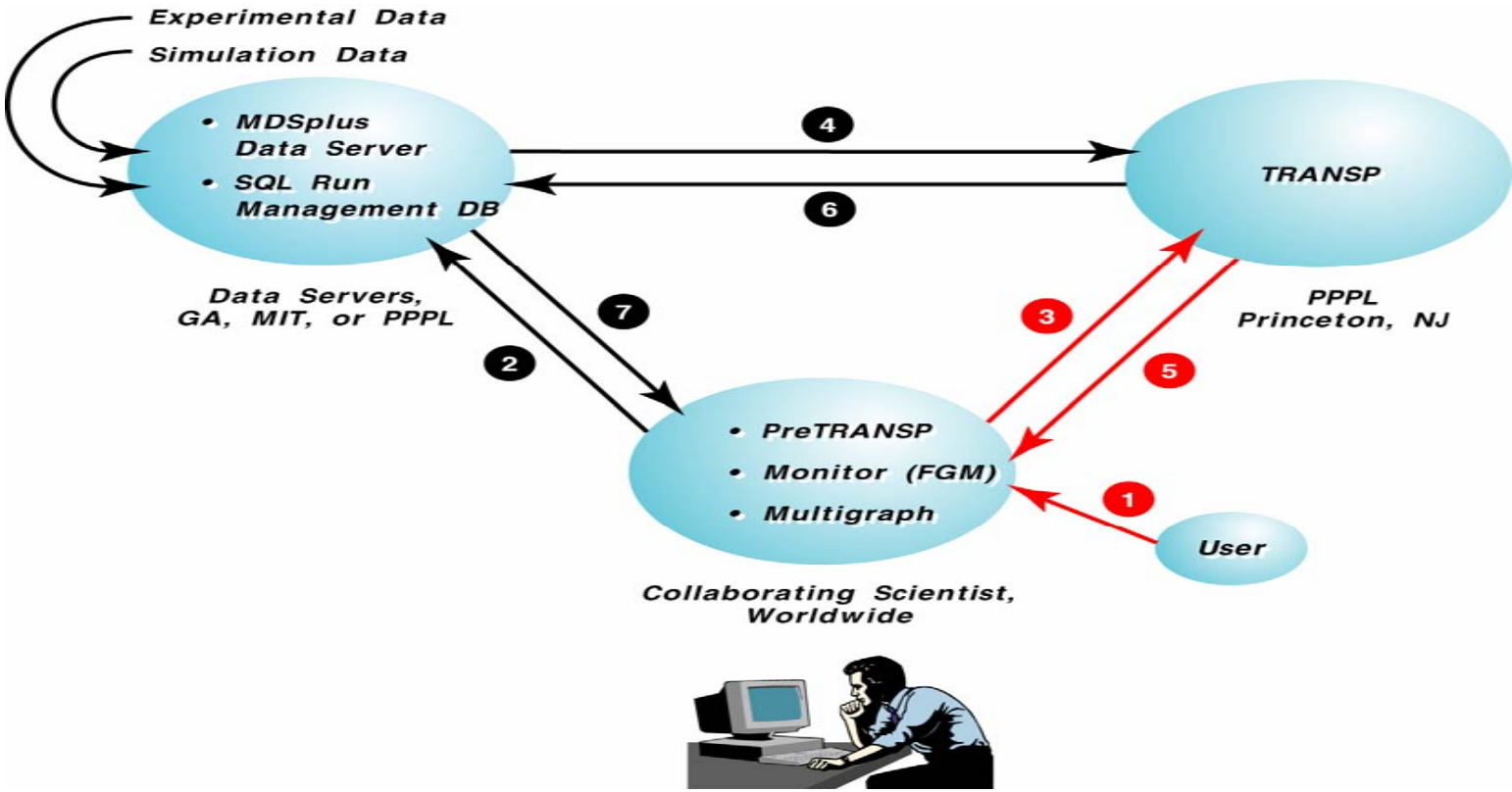
MONITORING THE NATIONAL FUSION GRID

- The Fusion Grid Monitor (FGM) tracks the state of applications on the FusionGrid
  - Real time status of individual code runs
  - Detailed logfile access for each code run
  - Code maintenance notification

- Users are notified with status updates from individual code runs:
  - Current state and owner of the grid run
  - CPU and Wall time used by run thus far
  - When the run was queued, started, and stopped (completion, or stopped by an error)
  - Where the application is being run, as well as logfiles associated with the grid run

- FGM currently tracks grid-enabled TRANSP on the FusionGrid
  - PreTRANSP on the local client
  - Grid-enabled TRANSP at PPPL
  - Centralized logfile access through anonymous FTP

TRANSP IMPLEMENTATION ON THE FUSIONGRID

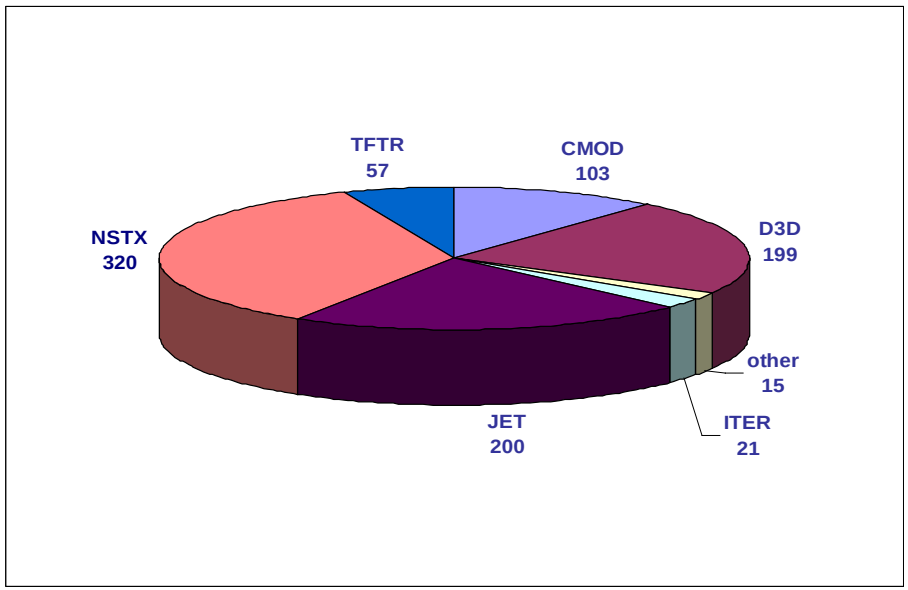


- PreTRANSP is used to launch grid-enabled TRANSP from a local client
- Grid Security and authorization / authentication is done through Akenti, and the Globus toolkit
- Grid-enabled TRANSP runs on a cluster at PPPL
- TRANSP data is loaded into MDSplus at GA, MIT, or PPPL
- FGM tracks the state of each code run as it goes through the process of running on the FusionGrid

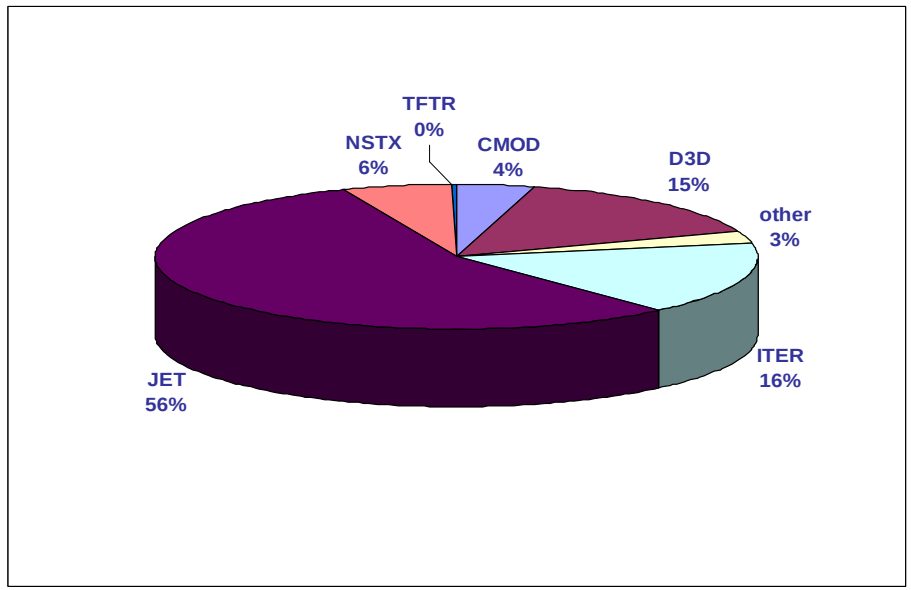
FGM LOGS STATUS UPDATES TO A RELATIONAL DATABASE

- FGM stores all status updates into a relational database at GA
  - This allows overview information on the use of applications on the FusionGrid
  - 915 TRANSP Runs: Oct 2002 – June 2003
  - 5962 CPU Hours: Oct 2002 – June 2003

Number of runs used by FusionGrid researchers / collaborators



Percentage of CPU hours used by FusionGrid researchers / collaborators



CHANGES MADE TO THE MONITOR FOR USE ON THE FUSIONGRID

- Logfile access was updated from using rcp, to use anonymous FTP
  - Logfile access for DAM was done through rcp, since logfiles are always located at the same site
  - For the FusionGrid, logfiles are located at multiple sites, so anonymous FTP is used to provide logfiles through FGM
  - Providing logfiles through anonymous FTP allows users to browse logfiles without knowing which machine they came from, and allows an indirect level of security, as it relies on administrators of each site to only post those logfiles they want available

- Asynchronous code run tracking.
  - The monitor was originally applied for use at DIII-D to track a linear shot cycle
  - On the FusionGrid however, multiple code runs need to be tracked concurrently
  - The monitor was updated to handle the needs of tracking asynchronous code runs

DAM AND FGM BOTH RUN ON A SMALL LINUX SYSTEM (NSSRV1.GAT.COM)

- About NSSRV1
  - NSSRV1 is a Netscape Server
  - NSSRV1 runs DAM and FGM at DIII-D
  - Login to the system is limited
  - Averages 420 MB Available Memory
  - Averages 0.2 load average

- NSSRV1 System Hardware
  - Intel Pentium III 733 Mhz
  - Linux RedHat 6.2
  - 1 GB RAM
  - 100 BaseT network connection
  - 30 GB hard disk

- Software Used by the Monitor
  - Java 1.3
  - JSDK 2.1
  - Jess 5.2
  - Sybase's dblib client
  - JDBC

- Quick References
  - Jess : http://herzberg.ca.sandia.gov/jess/
  - Java / JDBC / JSDK : http://java.sun.com/
  - Sybase : http://sybase.com/
  - DAM / FGM : General Atomics, Sean Flanagan
    flanagan@fusion.gat.com