A Scalability Model for ECS's Data Server

Daniel A. Menascé
Dept. of Computer Science
George Mason Univ., MS 4A5
Fairfax, VA 22030
menasce@cne.gmu.edu
tel: +1 703 993 1537
fax: +1 703 993 1710

Mukesh Singhal
Dept. of Computer and Info. Sc.
The Ohio State Univ.
Columbus, OH 43210
singhal@cis.ohio-state.edu
tel: +1 614 292 5839
fax: +1 641 292 2911

Abstract

This paper presents a model for the scalability analysis of the Data Server subsystem of the EOSDIS Core System (ECS). The goal of the model is to analyze to determine if the planned architecture of the Data Server will support an increase in the workload with the possible components upgrade. We provide a summary of the ECS's Data Server architecture and of a high level description of the Ingest and Retrieval operations in the ECS's Data Server. This description forms the basis for the development of our scalability model. Then we present details of the scalability model and the methodology used to solve it. We describe the structure of the scalability model, input parameters, expressions for computing parameters of the scalability model solver, and algorithms for solving the scalability model. The scalability model is very general and allows the modeling of data servers with numerous configurations.

1 Introduction

Perhaps one of the most important examples of large-scale, data-intensive, geographically distributed, information systems is NASA's Earth Observing System (EOS) Data and Information System (EOSDIS). EOS is a NASA mission aimed at studying the planet Earth. A series of satellites with scientific instruments aboard will be collecting important data about the Earth's atmosphere, land, and oceans over a period of 15 years. This mission will generate an estimated tera bytes/day of raw data which will be processed to generate higher level data products [2]. Raw data received from the satellites is first stored as Level 0 (L0) data which may then be transformed after successive processing into levels 2 through 4 (L2 - L4). Data received from the satellites and the data products generated from them will be stored at various Distributed Active Archive Centers (DAACs) located throughout the United States. An important component of a DAAC is the Data Server—the subsystem that stores and distributes data as requested by EOSDIS users.

The Data Server stores its information using a hierarchical mass storage system that uses a combination of automated tape libraries and disk caches to provide cost-effective

storage for the large volumes of data held by the Data Server. Performance studies and workload characterization methods and software for hierarchical mass storage systems are reported in [3, 5, 6, 7, 8].

In this paper, we present a model for the scalability analysis of the Data Server subsystem of the EOSDIS Core System (ECS). The goal of the model is to analyze if the planned architecture of the Data Server will support an increase in the workload with the possible upgrade and/or addition of processors, storage subsystems, and networks. This analysis does not contemplate new architectures that may be needed to support higher demands.

The remaining sections of this paper are organized as follows. Section 2 provides a summary of the architecture of ECS's Data Server as well as a high level description of the Ingest and Retrieval operations as they relate to ECS's Data Server. This description forms the basis for the development of the scalability model of the data server. Section 3 presents the scalability model and the methodology used to solve it. This section describes the structure of the scalability model, input parameters, algorithms for computing parameters of the scalability model solver, algorithms for solving the scalability model, and the assumptions and rationale behind these assumptions. The scalability model takes into account the proposed hardware and software architecture. The model is quite general and allows the modeling of data servers with numerous configurations.

2 Ingest and Retrieval Operations

This section provides a high level description of the Ingest and Retrieval workloads of the ECS's Data Server. This description forms the basis for the development of a model to analyze the scalability of the Data Server. The scalability analysis entails determining whether the current architecture of the ECS Data Server supports an increase in the workload intensity with possibly more processing and data storage elements of possibly higher performance.

2.1 Subsystems of the Data Server

The following subsystems of the Data Server will be considered for the purpose of the scalability analysis considered in this study:

Software Configuration Items:

- **Science Data Server (SDSRV):** responsible for managing and providing access to non-document earth science data.
- **Storage Management (STMGT):** stores, manages, and retrieves files on behalf of other SDPS components.

Hardware Configuration Items:

- **Access Control and Management (ACMHW):** supports the Ingest and Data Server subsystems that interact directly with users. Of particular interest here is the SDSRV.

- **Working Storage (WKSHW):** provides high performance storage for caching large volumes of data on a temporary basis.
- **Data Repository (DRPHW):** provides high capacity storage for long-term storage of files.
- **Distribution and Ingest Peripherals (DIPHW):** supports ingest and distribution via physical media.

2.2 Ingest Data Operation

The diagram in Figure 1 depicts the flow of control and data for the Ingest process. We have not included Document Repository nor the Document Data Server due to their small impact on scalability if compared with ingest of L0 data. Circles in the diagram represent processes. The labels in square brackets beside each process indicate the hardware configuration item they execute on. Bolded labels indicate hardware configuration items that belong to the Data Server.

The main aspects of the diagram of figure 1 are discussed below:

- Incoming L0 data is first stored into the system on the Staging Disk, and then into AMASS's cache—the hierarchical mass storage systems' disk cache for files. The metadata are extracted and entered into a Metadata database managed by Sybase and the actual data are archived. This is depicted in figure 2.
- The SDPF (Science Data Processing Function) represents the users of the Ingest system and negotiates with the Ingest Request Manager for coordination of transferring data into the Ingest system.
- Data are initially entered through an interactive GUI interface, or, most of the time from external data providers through ftp or direct transfer of files, if that is done on the same local network, into the Staging Disk.
- The actual data is then transferred into AMASS' disk cache. From the cache, the data migrates to robotically mounted tapes managed by AMASS. The metadata extracted from the data is stored into a Metadata Database managed by Sybase.
- The SDSRV (Science Data Server) gives the metadata templates to the Ingest Request Manager for it to extract metadata.
- There are two and sometimes three SDSRV's and one STMGT (Storage Management) processes. The Ingest Request Manager process selects which SDSRV to use.

The scalability analysis will among other things determine possible performance bottlenecks. The staging disk, the AMASS disk cache, and the metadata extraction process are likely candidates for bottlenecks.

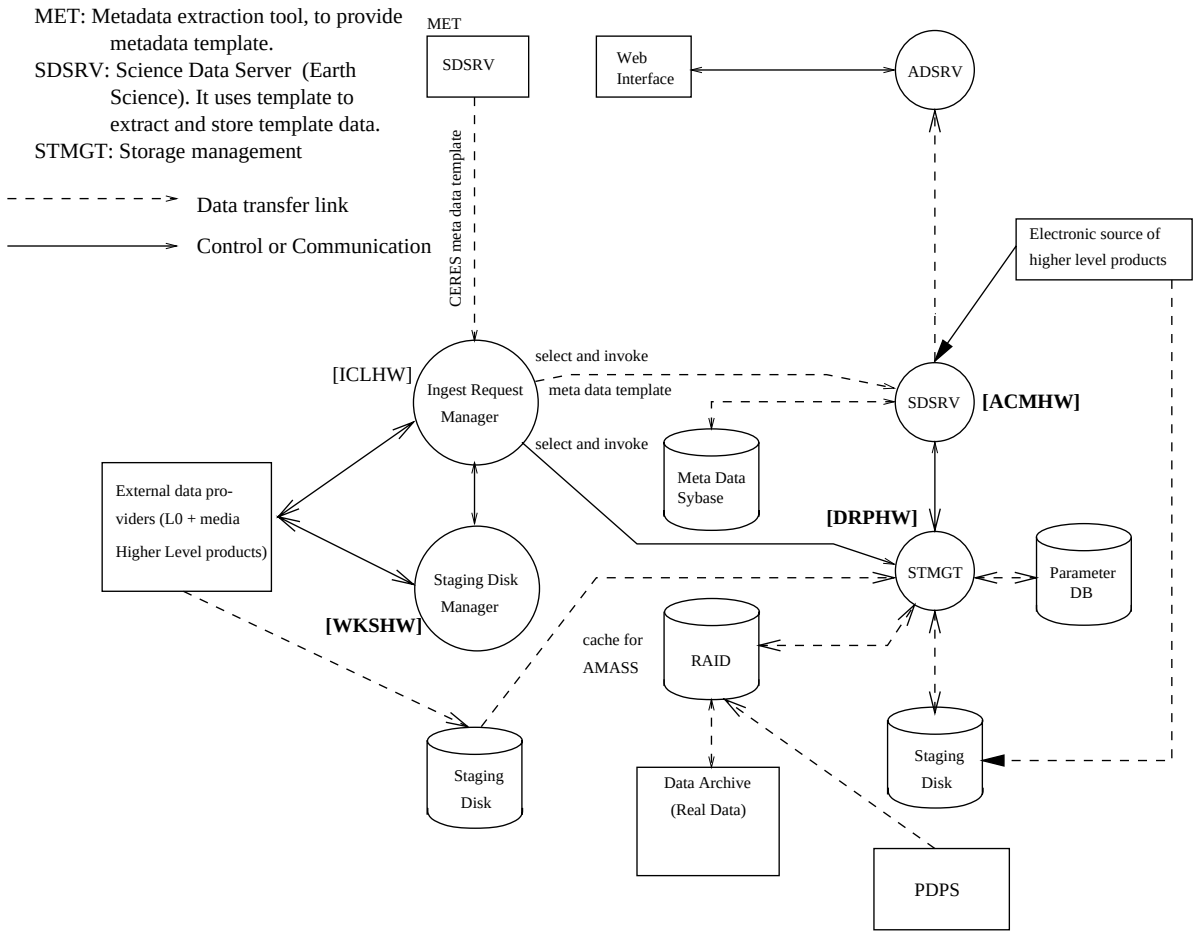

Figure 1: L0 Ingest Control and Data Flow.

2.3 Retrieval Operation

This section examines the retrieval and processing operation on L1+ data. Figure 3 depicts the flow of control and data for this operation. Circles in the diagram represent processes. The labels in square brackets beside each process indicate the hardware configuration item they execute on. Bolded labels indicate hardware configuration items that belong to the Data Server.

The retrieval operation proceeds in the following three stages:

Stage 1: Checking data and deciding what processing is required:

- SDSRV initiates the retrieval process by notifying the Subscription Server of the new data arrival.
- The Subscription Server performs a subscription checkfor this data and performs an appropriate notification, e.g., email notification, etc.

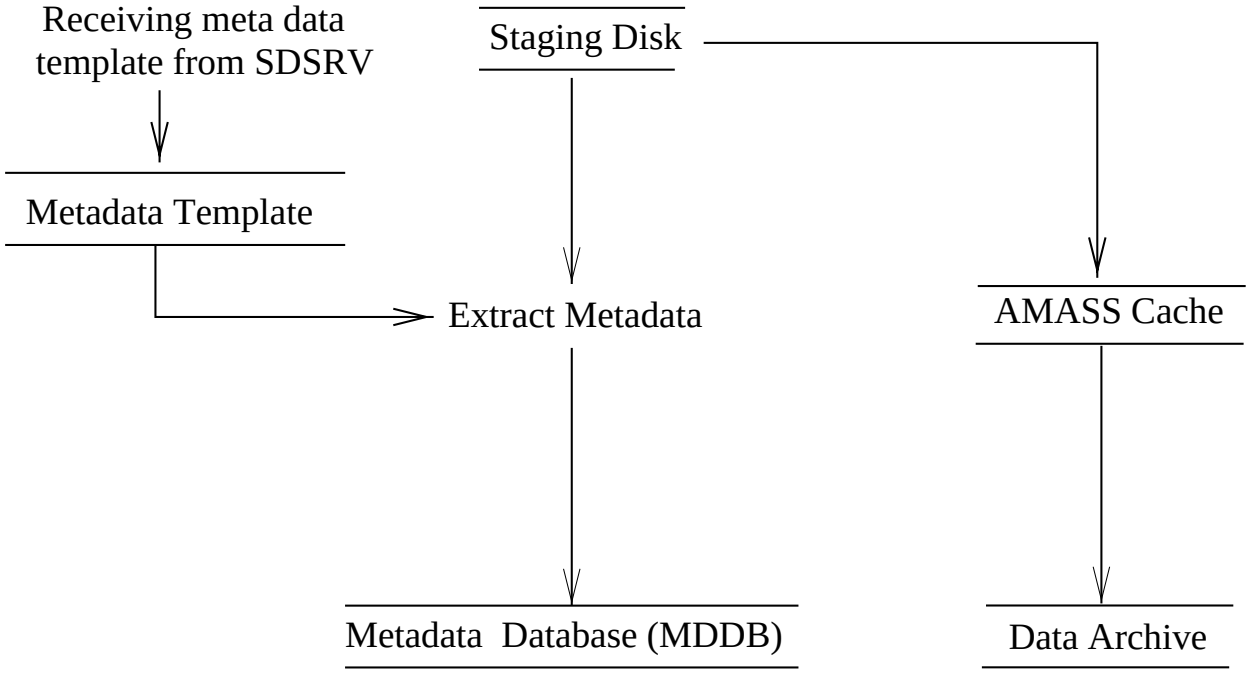

Figure 2: Data Flow Diagram for Ingest Data.

- The Subscription Server notifies PDPS PLANG of new data arrival.
- PLANG figures out (e.g., retrieves) a processing plan and based on the processing plan, passes the processing request to PRONG.
- PDPS PRONG connects to the appropriate SDSRV (may not be the SDSRV which initiated the retrieval and processing operations).

Stage 2: Retrieving data:

- The SDSRV requests that the Data Distribution Services CSCI (DDIST) retrieves the data files.
- SDSRV $\longrightarrow^{requests}$ DDIST $\longrightarrow^{requests}$ STMGT. The STMGT retrieves the files from AMASS archive into the AMASS cache if it is already not present in the cache.
- SDSRV notifies PRONG of data (identified by UR) availability.

Stage 3: Processing data and archiving, both data and metadata:

- PRONG transfers the retrieved data from the Working Storage to local PDPS disk. (If the AMASS cache and Working Storage are on different devices, then data must be first moved from the former to the latter.)
- PRONG processes the retrieved data to produce a higher level product.

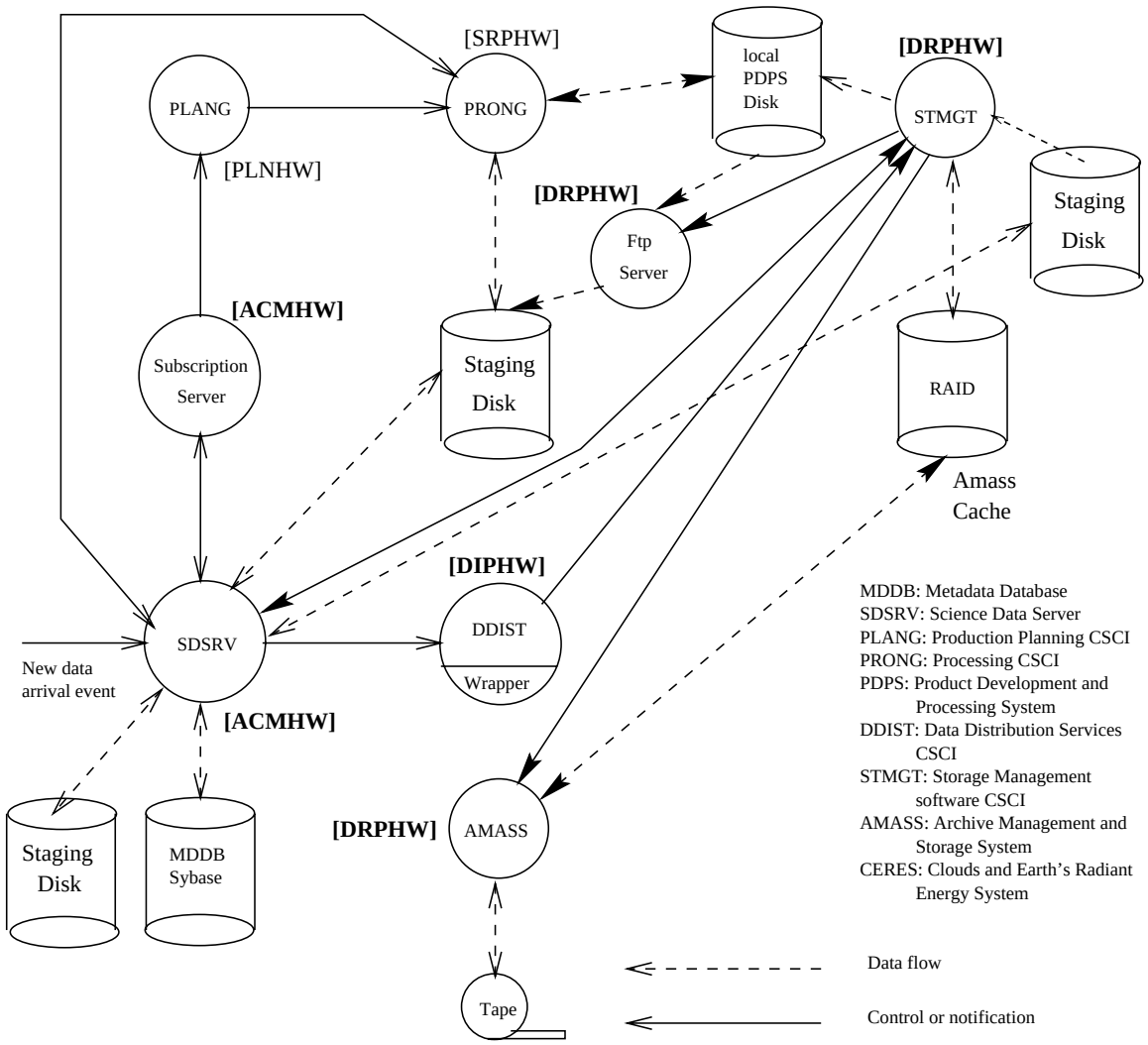

Figure 3: A Flow Diagram of Data Retrieval and Processing

- PRONG processes the data to a higher-level product and extracts metadata from the higher-level data using the Metadata Extraction Tool and populates the target metadata template and writes a metadata file (on MDDB Sybase).
- PDPS PRONG sends an insert request to SDSRV.
- SDSRV $\longrightarrow^{requests}$ STMGT $\longrightarrow^{requests}$ AMASS. The AMASS file manager archives the files. Archiving is done in two steps:
  - STMGT copies data from PDPS (local disk) to Working Storage via an ftp command.
  - data are copied from the Working Storage to AMASS cache (and then to AMASS archive).

- SDSRV inserts metadata in the Metadata Database (MDDB) and then notifies PRONG that the archival operation has been completed.

2.4 Assumptions

The various software processes shown in the previous subsection were mapped into the different hardware configuration items for the GSFC, EDC, and LaRC DAACs. The following assumptions were made when developing the scalability model.

- Processing of "Ingest data" and "Data retrieval and processing" constitute the main load on the Data Server. Thus, we modelled only these two operations.
- We did not model users' requests for data to be subsetted or subsampled nor we modelled compressed data.
- In data retrieval operations, PLANG retrieves a processing plan from a database (e.g., Sybase).
- The AMASS cache and the working storage may be implemented on the same disk.
- Servers that are not potential bottlenecks were not considered in the model. Examples include the "subscription server" and PDPS.
- We assume that mean arrival rate of both types of requests (ingest data and data retrieval) and service demands of these requests at various service stations are available or can be easily estimated.

3 A Scalability Model

We now describe our scalability model for the ECS's Data Server and our methodology for solving this model. We describe the structure of the scalability model, input parameters, algorithms for computing parameters of the scalability model solver, and algorithms for solving the scalability model. We describe our assumptions and rationale for these assumptions.

The scalability model is based on our understanding of the architecture of ECS's Data Server and the Ingest and Retrieval operations described in the previous section. The sole purpose of the model is to analyze the scalability of the Data Server, i.e., to determine whether the current architecture of the ECS Data Server can support an increase in the workload intensity.

3.1 A Framework for Scalability Analysis

Figure 4 gives the structure of the scalability model. The "Scalability Model Generator" collects information from three input files (these files define the modeling information on the ECS's data server and the workload) and processes this information to create an output file which contains inputs to the "Scalability Model Solver". This solver uses queuing

network [4] techniques to obtain desired performance measures such as response times per workload, device utilizations, bottleneck indications, and queue lengths.

The first input file to the Scalability Model Generator, "Hardware Objects", defines the hardware resources (e.g., processors, disks, networks, and tape libraries) of the Data Server. The second input file to the Scalability Model Generator, "Workloads and Execution Flow", completely characterizes the workload that drives the Data Server. The third input file to the Scalability Model Generator, "Processes", defines the parameters of the software modules that will be executed on hardware servers by arriving requests for service (i.e., the workload).

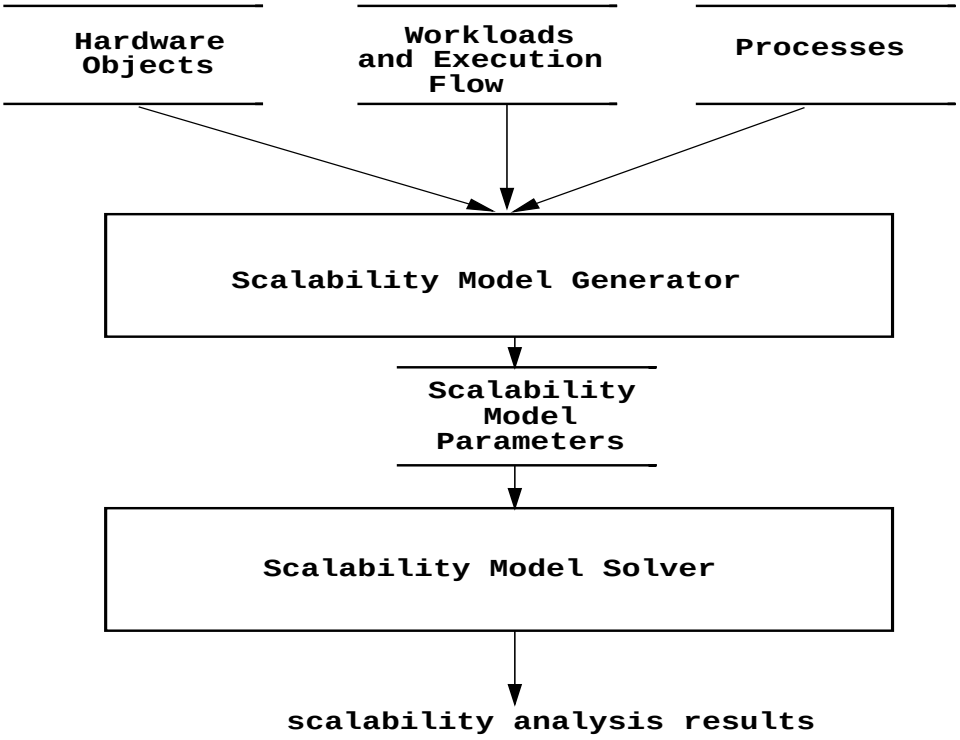

Figure 4: Scalability Model Framework.

The Scalability Model Generator reads information in these three files, processes this information, and generates an output file that contains the service demands for every resource in the queuing network model of Service demand is the total service time of a request of a certain workload type at a given device. The service demand does not include any time waiting to get access to the device. Waiting times are obtained by solving the model. The equations that form the basis of computation of service demands are presented in Section 3.3. The Scalability Model Solver reads information about the service demand from this file and solves the queuing network model for desired performance measures. The underlying equations that form the basis for a solution are described in Section 3.4.

3.2 Parameters for the Scalability Model

The parameters used in the scalability model are:

- $\mathcal{P}$: set of processes
- NCPUs_s: number of processors of server s
- SPint_s: SPECint95 rating of server s
- SPfp_s: SPECfp95 rating of server s
- TypeSP_p: type (e.g., int or fp) of the SPEC rating used to specify the computation demand for process p.
- SP_p: SPEC rating of the machine used to measure the computation demand of process p.
- $\mathrm{ComputeDemand}_p$: compute demand of process p measured at a machine with SPEC rating SP_p, in seconds
- $\mathrm{PExec}_{p,w}$: probability that process p executes in workload w
- $\mathrm{Seek}_{d,s}$: average seek time of single disk d of server s, in seconds
- $\mathrm{Latency}_{d,s}$: average rotational latency of single disk d of server s, in seconds
- $\mathrm{TransferRate}_{d,s}$: transfer rate of single disk d of server s, in MBytes/sec
- $\mathrm{Hit}_{da,s}$: cache hit ratio for disk array d
- $\mathrm{RAIDSeek}_{da,s}$: average seek time at any of the disks that compose disk array da at server s, in seconds
- $\mathrm{RAIDLatency}_{da,s}$: average rotational latency at any of the disks that compose disk array da at server s, in seconds
- $\mathrm{RAIDRate}_{da,s}$: transfer rate of any of the disks that compose disk array da at server s, in Mbytes/sec
- $\mathrm{NTDrives}_{t,s}$: number of tape drives of tape library t at server s.
- $\mathrm{NRobots}_{t,s}$: number of robots of tape library t at server s.
- $\mathrm{Rewind}_{i,t,s}$: rewind time of tape drive i of tape library t at server s.
- $\mathrm{MaxTSearch}_{i,t,s}$: maximum search time of tape drive i of tape library t, in seconds at server s.
- $\mathrm{TapeRate}_{i,t,s}$: transfer rate of tape drive i of tape library t at server s, in Mbytes/sec
- $\mathrm{Exchanges}_{t,s}$: number of tape exchanges per hour for each robot of tape library t at server s. (Each exchange involves putting the old tape in the tape library and loading the new tape into the tape drive.)

- $\mathrm{FilesPerMount}_{p,t,s}$: average number of files accessed per mount by process p at tape library t at server s.
- $\mathrm{FileSizePerMount}_{p,t,s}$: average size of files accessed by process, in Kbytes p per mount at tape library t at server s.
- $\mathrm{Bandwidth}_n$: bandwith of network n, in Mbps
- NType_n: type of network n.
- $\mathrm{NumBlocks}_{d,s,p}$: number of blocks accessed by process p at single disk d at server s.
- $\mathrm{BlockSize}_{d,s,p}$: block size for each access to single disk d at server s by process p, in KBytes
- $\mathrm{NumBlocksRead}_{da,s,p}$: number of blocks read by process p from disk array da at server s
- $\mathrm{NumBlocksWritten}_{da,s,p}$: number of blocks written by process p to disk array da at server s
- $\mathrm{StripeUnitSize}_{da,s}$: size of the stripe unit for disk array da at server s, in Kbytes
- Server_p: server in which process p is allocated
- λ_w: arrival rate of workload w, in requests/sec
- $\mathcal{P}_w$: set of processes executed by workload w
- $\pi_w = \{(p, x) \mid p \in \mathcal{P} \text{ and } x = Pr[p \text{ is executed in workload } w\}$: process flow within workload w.
- $\mathrm{PNet}_{n,w}$: probability that network n is traversed by workload w.
- $\mathrm{Volume}_{n,w}$: total data volume transferred through network n by workload w, in Kbytes

The input parameters for the Scalability Model Solver are:

- $D_{i,p,w}$: average service demand of process p in workload w at device i, i.e., the total time spent by the process at the device for workload w. This time does not include any queuing time.
- λ_w: average arrival rate of requests of workload w that arrive to ECS's Data Server.

3.3 Algorithms for Computing the Scalability Model Solver Parameters

In this section, we derive expressions for computing service demands for workloads at various types of devices. The service demand at a device due to a task is defined as the multiplication of the visit count of the task to the device and the service time of the task per visit to the device. The service demand represents the total average time spent by the task at the device.

3.3.1 Computation of Service Demands for Processors

The service demand that a task in workload w presents at a server s due to the execution of a process p is given by:

$$D_{s,p,w} = \frac{\text{ComputeDemand}_p \times \text{PExec}_{p,w}}{\text{ScaleFactor}(p,s)} \tag{1}$$

where

$$\text{ScaleFactor}(p,s) = \begin{cases} \text{SPint}_s / SP_p & \text{if TypeSP}_p = \text{int} \\ \text{SPfp}_s / SP_p & \text{if TypeSP}_p = \text{fp} \end{cases} \tag{2}$$

Since ComputeDemand_p is given for a processor of certain rating, $\text{ScaleFactor}(p,s)$ is used to normalize the process service time to the speed-rating of the current processor.

The service demand, $D_{s,w}$, of a workload w at the CPU of server s is then

$$D_{s,w} = \sum_{\forall p \in \mathcal{P}_w \mid s = \text{Server}_p} D_{s,p,w} \tag{3}$$

3.3.2 Computation of Service Demands for Single Disks

The service demand that a task in workload w presents to a disk d at a server s due to the execution of a process p is given by:

$$\begin{aligned} D_{d,s,p,w} &= \text{PExec}_{p,w} \times \text{NumBlocks}_{d,s,p} \times \\ &\quad \left[\text{Seek}_{d,s} + \text{Latency}_{d,s} + \frac{\text{BlockSize}_{d,s,p}}{\text{TransferRate}_{d,s} \times 1000} \right] \end{aligned} \tag{4}$$

The term "$\text{Seek}_{d,s} + \text{Latency}_{d,s} + \frac{\text{BlockSize}_{d,s,p}}{\text{TransferRate}_{d,s} \times 1000}$" denotes the time the disk takes to fetch one block of data.

The service demand, $D_{d,s,w}$, of a workload w at disk d of server s is then

$$D_{d,s,w} = \sum_{\forall p \in \mathcal{P}_w \mid s = \text{Server}_p} D_{d,s,p,w} \tag{5}$$

3.3.3 Computation of Service Demands for Disk Arrays

The computation of service demands for disk arrays is involved and is done in several steps. The number of blocks that a process p reads at a disk (i.e., the number of stripe accesses) in disk array da at server s is given by

$$\text{NumBlocksReadPerDisk}_{da,s,p} = \left\lceil \frac{\text{NumBlocksRead}_{da,s,p} \times \text{BlockSize}_{d,s,p}}{5 \times \text{StripeUnitSize}_{da,s}} \right\rceil \tag{6}$$

where $\lceil x \rceil$ denotes the ceiling of x. The numerator denotes the total volume of information read from all five disks in the disk array and the denominator denotes the volume of information read from all five disks in a single stripe group access.

The service time to process each stripe request at each disk is given by the following equation: (The first subexpression indicates that the seek time is amortized over all stripe unit accesses.)

$$\begin{aligned}\text{ServiceTimePerDisk}_{da,s,p} &= \frac{\text{RAIDSeek}_{da,s}}{\text{NumBlocksReadPerDisk}_{da,s,p}} + \text{RAIDLatency}_{da,s} + \\ &\quad \frac{\text{StripeUnitSize}_{da,s}}{\text{RAIDRate}_{da,s}}\end{aligned} \tag{7}$$

The service demand due to read requests at a disk in disk array da at server s due to execution of process p in workload w is given by the following equation: (Since a disk array has a data cache, term $(1 - \text{Hit}_{da,s})$ denotes the probability that data to be read is not available in the cache and a read access will have to be made.)

$$\begin{aligned}\text{ReadServiceDemandPerDisk}_{da,s,p,w} &= \text{NumBlocksReadPerDisk}_{da,s,p} \times \\ &\quad \text{ServiceTimePerDisk}_{da,s,p} \times \\ &\quad \text{PExec}_{p,w} \times (1 - \text{Hit}_{da,s})\end{aligned} \tag{8}$$

Now the service demand, $D^r_{da,s,p,w}$, due to read requests at disk array da at server s due to execution of process p in workload w is given by the following equation:

$$D^r_{da,s,p,w} = \frac{H_5 \times \text{ReadServiceDemandPerDisk}_{da,s,p,w}}{1 - \text{USingleDisk}_{da,s,p,w}} \tag{9}$$

where $H_5 = \sum_{j=1}^{5} 1/j = 2.28$ and $\text{USingleDisk}_{da,s,p,w}$ is given by Eq. (14). The term H_5 shows up in the expression because a read request at the disk array is complete only after the last read at its disks is done. This approximation is based on [5].

The service demand, $D^r_{da,s,w}$, of a workload w at the disk array da of server s is then

$$D^r_{da,s,w} = \sum_{\forall p \in \mathcal{P}_w \mid s = \text{Server}_p} D^r_{da,s,p,w} \tag{10}$$

The computation of the service demand due to write requests at disk array da at server s due to the execution of process p in workload w is similar. The computation of the number of blocks that a process p writes at a disk (i.e., the number of stripes written) in the disk array da at server s is somewhat different and is given by the following equation: (The (4/5) term in the denominator is due to the fact that a parity block is generated for every four blocks written onto the disks. Thus 25% additional data is generated.)

$$\text{NumBlocksWrittenPerDisk}_{da,s,p} = \left\lceil \frac{\text{NumBlocksWritten}_{da,s,p} \times \text{BlockSize}_{d,s,p}}{(4/5) \times \text{StripeUnitSize}_{da,s}} \right\rceil \tag{11}$$

$$\begin{aligned}\text{WriteServiceDemandPerDisk}_{da,s,p,w} &= \text{NumBlocksWrittenPerDisk}_{da,s,p} \times \\ &\quad \text{ServiceTimePerDisk}_{da,s,p} \times \\ &\quad \text{PExec}_{p,w} \end{aligned} \tag{12}$$

$$D^{w}_{da,s,p,w} = \frac{H_5 \times \text{WriteServiceDemandPerDisk}_{da,s,p,w}}{1 - \text{USingleDisk}_{da,s,p,w}} \tag{13}$$

where

$$\begin{aligned}\text{USingleDisk}_{da,s,p,w} &= \text{PExec}_{p,w} \times \lambda_w \times \\ &\quad [(\text{NumBlocksReadPerDisk}_{da,s,p} + \\ &\quad \text{NumBlocksWrittenPerDisk}_{da,s,p}) \times \\ &\quad \text{ServiceTimePerDisk}_{da,s,p}] \end{aligned} \tag{14}$$

The service demand, $D^{w}_{da,s,w}$, of a workload w at the disk array da of server s is then

$$D^{w}_{da,s,w} = \sum_{\forall p \in \mathcal{P}_w \mid s = \text{Server}_p} D^{w}_{da,s,p,w} \tag{15}$$

3.3.4 Computation of Service Demands for Tape Libraries

The computation of the service demands for tape drives and robots at a tape library is involved and is done in several steps.

The total average seek time that a process p experiences at tape drive i in tape library t at server s is given by the following equation: (The factor "1/2" is due to the fact that the first file access will result in searching half the tape on the average and the factor "1/3" shows up because the remaining file accesses will require searching 1/3 of the tape on the average.)

$$\text{AverageSeekTime}_{t,s} = \text{MaxTSearch}_{i,t,s} \times [1/2 + (\text{FilesPerMount}_{p,t,s} - 1)/3] \tag{16}$$

The average tape mount time in seconds at tape drive i in tape library t at server s is given by

$$\text{MountTime}_{i,t,s} = 3,600/2 \times \text{Exchanges}_{t,s} \tag{17}$$

The time that tape drive i in tape library t at server s takes to serve a file access request is given by

$$\begin{aligned}\text{TapeDriveServiceTime}_{i,t,s} &= \text{AverageSeekTime}_{t,s} + \\ &\quad \frac{\text{FilesPerMount}_{p,t,s} \times \text{FileSizePerMount}_{p,t,s}}{\text{TapeRate}_{i,t,s}} + \\ &\quad \text{Rewind}_{i,t,s} \end{aligned} \tag{18}$$

The average robot service time is then

$$\mathrm{RobotServiceTime}_{t,s} = 2 \times \mathrm{MountTime}_{i,t,s} \tag{19}$$

So, the service demand at the tape drive i of tape library t of server s due to the execution of process p in workload w is

$$D^{tape\ drive}_{i,t,s,p,w} = \mathrm{PExec}_{p,w} \times \mathrm{TapeDriveServiceTime}_{i,t,s}/\mathrm{NTDrives}_{t,s} \tag{20}$$

The service demand at the tape drive i of tape library t of server s due to workload w is

$$D^{tape\ drive}_{i,t,s,w} = \sum_{\forall p \in \mathcal{P}_w \mid s=\mathrm{Server}_p} D^{tape\ drive}_{i,t,s,p,w} \tag{21}$$

The average service demand at any robot of tape library t of server s due to the execution of process p in workload w is

$$D^{robot}_{t,s,p,w} = \mathrm{PExec}_{p,w} \times \mathrm{RobotServiceTime}_{t,s}/\mathrm{NRobots}_{t,s} \tag{22}$$

The service demand at any robot of tape drive i of tape library t of server s due to workload w is

$$D^{robot}_{t,s,w} = \sum_{\forall p \in \mathcal{P}_w \mid s=\mathrm{Server}_p} D^{robot}_{t,s,p,w} \tag{23}$$

3.3.5 Computation of Service Demands for Networks

The service demand of workload w presents at network n is given by the following equation: (The term "$Volume_{n,w}/Bandwidth_n$" denotes the time taken by the network to transfer the data for a task in workload w.)

$$D^{network}_{n,w} = \frac{\mathrm{PNet}_{n,w} \times \mathrm{Volume}_{n,w} \times 8}{\mathrm{Bandwidth}_n \times 1000} \tag{24}$$

3.4 The Scalability Model

The scalability model uses queuing network (QN) models to determine the degree of contention at each of the devices that compose ECS's Data Server. The QN model used in this case is a multiclass open QN [4] with additional approximations to handle the case of disk arrays and to handle the instances of simultaneous resource possession that appear when modeling automated tape libraries [3]. The QNs used also allow for load dependent devices. Load dependent devices are used in the model to handle the following situations:

- Symmetric multiprocessors: this case is characterized by a single queue for multiple servers. In this case, the service rate $\mu(k)$ of the CPU as a function of the number of requests k is given by $k.\mu$ for $k \leq J$ and $J.\mu$ for $k > J$ where J is the number of CPUs and μ is the service rate of each CPU.

- Collision-based LANs: in this case, the throughput of the LAN decreases as the load increases due to an increase in the number of collisions. This can be modeled by using an appropriate service rate function $\mu(k)$ as a function of the load on the network [1].

An open multiclass QN is characterized by the number R of classes, the number K of devices, by a matrix $D = [D_{i,r}]\ i = 1, \cdots K,\ r = 1, \cdots, R$ of service demands per device per class, and by a vector $\vec{\lambda} = (\lambda_1, \cdots, \lambda_R)$ of arrival rates per class. For each device, one has to indicate its type. The following types of devices are allowed in the QN model:

- Delay devices: no queues are formed at these devices.
- Queuing Load Independent (LI) devices: queues are formed at these devices but the service rate of the device does not depend on the number of requests queued for the device.
- Load Dependent device (LD): queues are formed at these devices but the service rate of the device depends on the number of requests queued for the device. In the case of load dependent devices, one has to provide the service rate multipliers (see [4]) for each value of the number of customers. In most cases—this is true for multiprocessors and collision-based LANs—the value of the service rate multipliers saturates very quickly with the number of requests. Therefore, we only need to provide a small and finite number of service rate multipliers for each LD device.
- Disk Array: this is a special type of device used to model disk arrays (see Fig. 5 for a depiction of this type of device).

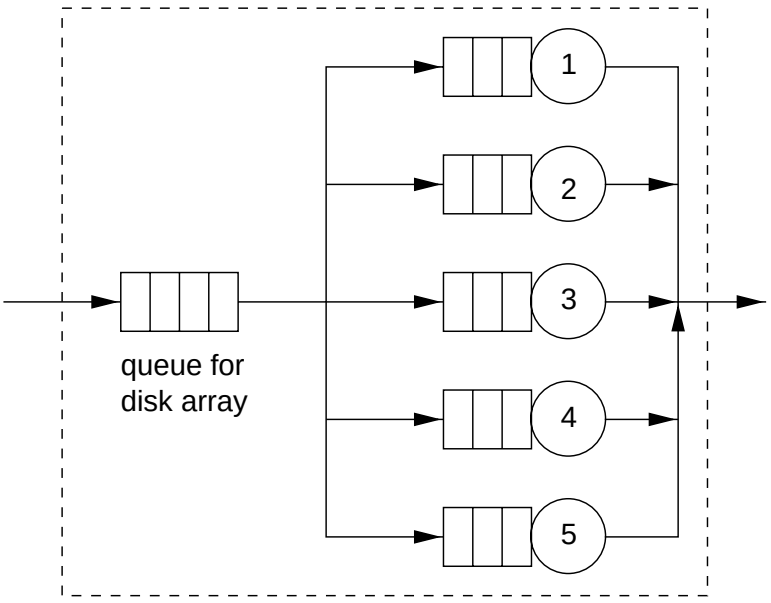

Figure 5: Disk Array.

The output results of an open multiclass QN are:

- $R'_{i,r}(\vec{\lambda})$: average residence time of class r requests at device i, i.e., the total time—including queuing and service—spent by requests of class r at device i.
- $R_r(\vec{\lambda})$: average response time of requests of class r. $R_r(\vec{\lambda}) = \sum_{i=1}^{K} R'_{i,r}(\vec{\lambda})$.

- $U_i(\vec{\lambda})$: utilization of device i.
- $n_{i,r}(\vec{\lambda})$: average number of requests of class r present at device i.
- $n_i(\vec{\lambda})$: average number of requests of at device i. $n_i(\vec{\lambda}) = \sum_{r=1}^{R} n_{i,r}(\vec{\lambda})$.

The basic equations for open multiclass QNs are (see [4]):

$$
\begin{aligned}
U_{i,r}(\vec{\lambda}) &= \lambda_r D_{i,r} \\
U_i(\vec{\lambda}) &= \sum_{r=1}^{R} U_{i,r}(\vec{\lambda}) \\
\bar{n}_{i,r}(\vec{\lambda}) &= \frac{U_{i,r}(\vec{\lambda})}{1 - U_i(\vec{\lambda})} \\
R'_{i,r}(\vec{\lambda}) &= \begin{cases} D_{i,r} & \text{delay device} \\ \dfrac{D_{i,r}}{1 - U_i(\vec{\lambda})} & \text{LI device} \end{cases} \\
R_r(\vec{\lambda}) &= \sum_{i=1}^{K} R'_{i,r}(\vec{\lambda}) \\
n_i(\vec{\lambda}) &= \sum_{r=1}^{R} n_{i,r}(\vec{\lambda})
\end{aligned}
$$

The extension to LD devices is given in [4].

4 Preliminary Results

The model presented here was applied to the Goddard Space Flight Center (GSFC) Data Server's architecture. Figure 6 displays the response time versus the arrival rate of data ingested in GB/day. The rate at which data is retrieved is assumed to be twice the ingest rate. It should be pointed out that at this point, these are very preliminary results and are not intended to reflect the performance of GSFC's Data Server. Data collection efforts are still in progress. The purpose of Figure 6 is to illustrate the type of results one can obtain from the type of models discussed in this paper. As it can be seen, in the curve, the ingest workload starts to saturate at approximately 160 GB/day while the retrieval workload supports a much higher data rate.

5 Concluding Remarks

In this paper, we derived the algorithms and expressions to be used to convert data describing the software and hardware architecture of ECS's Data Server into a scalability model. The model will be used to verify how well the Data Server supports an increase in workload intensity while maintaining reasonable performance. The scalability model is based on queuing network models that are automatically generated from the description of the architecture and the workload.

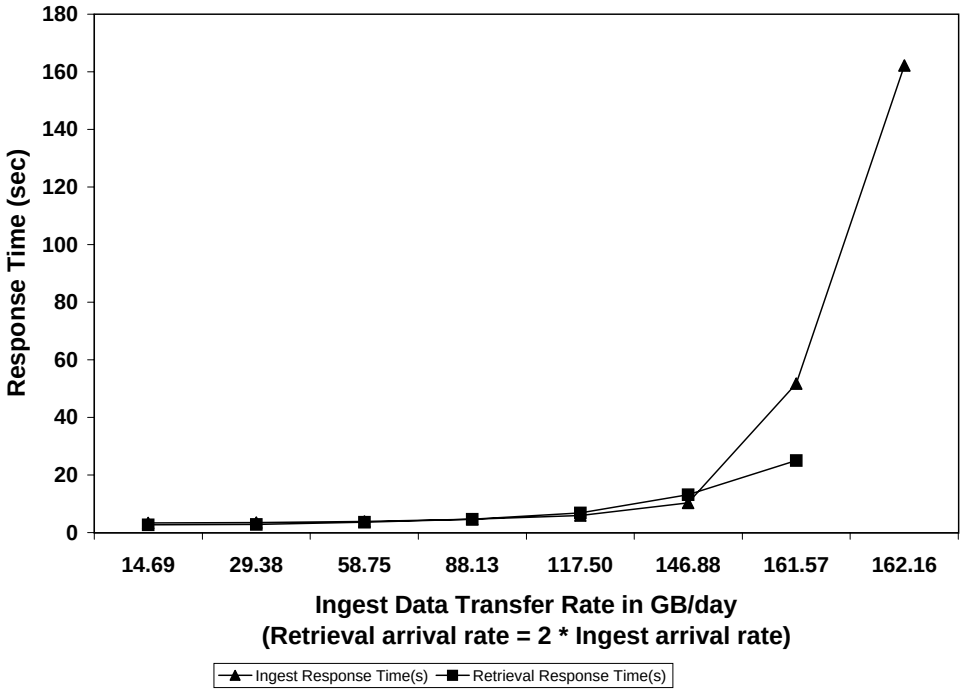

Figure 6: Response time versus data rate for GSFC.

References

[1] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queuing Network Models*, Prentice Hall, Englewood Cliffs, N. J., 1984.

[2] B. Kobler, J. Berbert, P. Caulk, and P. C. Hariharan, *Architecture and Design Storage and Data Management for the NASA Earth Observing System Data and Information System (EOSDIS)*, Fourteenth IEEE Symposium on Mass Storage Systems (Second International Symposium), Monterey, CA, September 11-14, 1995.

[3] D. A. Menascé, O. I. Pentakalos, and Y. Yesha, An Analytic Model of Hierarchical Mass Storage Systems with Network-Attached Storage Devices, *Proc. of the 1996 ACM Sigmetrics Conference*, Philadelphia, PA, May 1996.

[4] D. A. Menascé, V. A. F. Almeida, and L. W. Dowdy, *Capacity Planning and Performance Modeling: from mainframes to client-server systems*, Prentice Hall, Upper Saddle River, NJ, 1994.

[5] O. I. Pentakalos, D. A. Menascé, M. Halem, and Y. Yesha, *Analytical Performance Modeling of Hierarchical Mass Storage Systems*, IEEE Transactions on Computers, Vol. 46, No. 10, pp. 1103-1118.

[6] O. I. Pentakalos, D. A. Menascé, and Y. Yesha, *Pythia and Pythia/WK: Tools for the Performance Analysis of Mass Storage Systems*, Software Practice and Experience, October 1997.

[7] O. I. Pentakalos and D. A. Menascé, *Automated Clustering Based Workload Characterization for Mass Storage Systems*, Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies, College Park, MD, September 17-19, 1996.

[8] O. I. Pentakalos , D. A. Menascé, Y. Yesha, and M. Halem, *An Approximate Performance Model of a Unitree Mass Storage System*, Fourteenth IEEE Symposium on Mass Storage Systems (Second International Symposium), Monterey, CA, September 11-14, 1995.