

Demonstration on Measuring and Monitoring the Performance of DIENST

SARANTOS KAPIDAKIS* CHRISTOS NIKOLAOU† JAKKA SAIRAMESH‡
SOTIRIOS TERZIS§

SEPTEMBER 1-3, 1997

1 Introduction

Dienst is a protocol and implementation that provides access to distributed, decentralized, multi-format document collections over the World-Wide Web. It was originally developed in 1992-95 for the ARPA-funded Computer Science Technical Reports project in the USA, and currently forms the technological basis of the Networked Computer Science Technical Reports Library (NCSTRL). NCSTRL defines itself not as a “project” of limited duration, but as an international organization dedicated to building a permanent digital library in computer science and to acting as a testbed for future repository architectures; The European Research Consortium for Informatics and Applied Mathematics (ERCIM) represents Europe on NCSTRL’s Steering Committee. Many ERCIM sites and industrial partners initiated the SAMOS project, which aims both to implement a European branch of the global NCSTRL collection and to adapt the Dienst protocol and implementation for use by research in industry in areas other than computer science. SAMOS has been identified by the G7 Global Marketplace for SMEs initiative as a pilot-project/testbed and is already listed in this initiative’s Web site. See: http://nii.nist.gov/g7/10_global_mp/testbeds/registered.html. Currently, there are more than 80 Computer Science Technical Report Libraries participating in NCSTRL.

1.1 Dienst architecture

The Dienst system deployed in NCSTRL currently handles search requests by embedding them in HTTP transactions – the protocol of the World-Wide Web. This implementation has important advantages: HTTP is ubiquitous, it can carry metadata, and it allows interactivity over all popular Web browsers. From a user standpoint, a report collection consists of a unified space of uniquely identified reports, each of which may be available in a variety of formats. Using publicly available Web clients, users can search the collection, browse, read, download, or print individual reports in any of their available

*Institute of Computer Science, Foundation for Research and Technology – Hellas, P.O Box 1385 Heraklion 71110, Greece. E-mail: sarantos@csi.forth.gr Tel: +30-81-391678, Fax: +30-81-391671. (contact person)

†Institute of Computer Science, Foundation for Research and Technology – Hellas, Heraklion 71110, Greece. E-mail: nikolau@csi.forth.gr

‡IBM T. J. Watson Research Center, Parallel and Distributed Systems, Hawthorne, NY 10532, USA. E-mail: jakka@cs.columbia.edu

§Institute of Computer Science, Foundation for Research and Technology – Hellas, Heraklion 71110, Greece. E-mail: terzis@csi.forth.gr

formats. The system also provides site administrators with tools for managing their report collections, including automated submission procedures, indexing tools, and format conversion tools.

1.2 DIENST vs. WWW

What distinguishes a Dienst-based system from a collection of ordinary Web servers is the notion of repository – a concept that has come to dominate discussions of future digital libraries. A repository is a distributed collection in a specific domain with a defined administrative policy. More than just a collection of files, a repository has structures both for clustering files into “documents” and “works” (to which one can attach metadata, terms and conditions, prices, authentication procedures, etc.) and for decomposing the files into component parts (pages and sections). These structures allow searching to operate at the most appropriate level of document granularity, and they allow the design of replication methods to support scalability. By definition, a repository also provides mechanisms for controlling access rights: by whom may a given document be viewed, modified, or downloaded.

A Dienst-based collection is managed by a set of interoperating Dienst servers distributed over the network. Each of these servers manages three basic library services: repositories of multi-format technical reports; indexes of technical reports collections and search engines for these indexes; user interfaces to provide front-end services for browsing, searching, and accessing the collections. A network of centralized Collection Servers provide directories of locations for all other services. These multiple Dienst servers interoperate to provide a logically integrated collection, even though the collection may physically be distributed over multiple sites. Each individual Dienst server “knows” about other sites, and visa-versa, by periodically polling the Collection Servers.

1.3 Issues and Problems

We outline some issues and problems:

1) A system administrator wishes to monitor the following:

Remote processing time: average time spent in processing a dienst request in D_2 and D_3 . This implies that every request submitted by D_1 has to be monitored at D_2 and D_3 from a certain start time. Note that this does not capture the average network delay. Variables capturing the average processing time at D_2 and D_3 have to be measured and stored in D_2 and D_3 respectively, and retrievable by the administrator.

Remote query: the average time taken by a query at D_2 and D_3 . This includes the round-trip network delay between D_1 and D_2 , and D_1 and D_3 , and the remote processing time; time spent in searching the database of documents in D_2 and D_3 respectively. Corresponding variables have to be measured and stored in D_1 .

WWW interface time: the average time spent in submitting requests through the WWW interface to D_1 (time spent in the stubs).

Operating System overheads: The average time spent by D_1 in forking processes for servicing requests.

A specific query (tracking): The time spent by the query at the servers needs to be monitored and correlated in a distributed fashion. This implies measuring the performance variables of the components that service the query at D_1 , D_2 and D_3 , and correlating the corresponding variables.

2) Load Balancing Issues:

Consider that there are two backup servers B_1 and B_2 that keep a backup of the indexes of D_1 , D_2 and D_3 . Assume that D_2 has been marked as “down” due to a network failure or because it has a recent history of not responding in time. Then D_1 could route requests either to B_1 or B_2 . For example, D_1 could submit the request to the least loaded backup server.

In order to perform load balancing, D_1 has to observe the current load of the B_1 and B_2 before submitting the request. This implies a simple way for D_1 to get the variables such as remote processing time or remote query time at B_1 and B_2 , and then make routing decisions. The decision could be made, for example, by using the the current value of the variables (current load), or their averages (average load), or their standard deviations.

2 Performance analysis

Because of the importance of the NCSTRL network and the Dienst development, the Computer Science Department of Cornell University cooperated with the Institute of Computer Science, Foundation for Research and Technology - Hellas (ICS/FORTH) and ICS/FORTH analyzed the performance of the current DIENST system.

We studied various performance parameters (setup delays, response time, lock-waiting time, network delays) occurring in the DIENST software running in a computer and because of the communication that is carried out over the network. We used both instrumentation and log file analysis for investigating the user response time of the running system.

2.1 Architecture: for Measuring and Monitoring

We use a simple client-server architecture model for performance management of the DIENST system. This model is based on ideas from SNMP based performance management in networks and distributed systems.

We have broken down the performance monitoring framework into three main design components:

1. Defining performance parameters for various software modules and components: Each DIENST server keeps a list of well-defined parameters, which it updates based on every DIENST request it processes.

2. Measuring and storing the parameters: Info base: The values of the various parameters will be measured upon request by the system administrator on an hourly or daily basis or on some other time-scale. A measurement process (daemon) will update the performance variables and store their current values, averages, and variances in a variable database.

3. Protocol to monitor the performance parameters: For system-administrators to monitor and debug the DIENST-Server behavior, a protocol is provided to retrieve performance variables on demand from the variable database. The protocol is simply an extension (a new verb) of the DIENST protocol. The variable database is managed by a process, called DMP or the database manager process. This simply returns the variables in the database. A better way is to directly contact DMP for the values of the variables.

We extend the DIENST protocol in order to provide dynamic load balancing capabilities:

Consider that D_1 wants to perform load-balancing of requests, then D_1 can use the DIENST protocol and get the current load of the servers (such as backup servers or other servers that replicate indexes) and then route requests. The advantage is that D_1 does not have to know a new protocol to talk to DMP. Regional Control: Consider the new configuration of the DIENST servers (regional index servers and metaservers). Each region can advertise the DIENST verb extensions to access the performance variables (parameters), and simply hide the details of the measurement mechanisms and the variable database.

We measure performance parameters and model in detail the behavior of a request from the start to the finish. For this, we use mathematical symbols to represent clearly the time spent in the modules of the DIENST system. These mathematical symbols are essentially the performance variables that we measure and store.

Consider that one wants to measure the time taken by a module of the DIENST server during operation. In order to do this, the start-time and the finish-time of this module has to be time-stamped. From this, the time taken by this module is simply the difference. We denote this time by a variable associated with the module. The time spent in a module depends on other modules that help in servicing a query.

We analyze the process by dividing it into many modules: Submit Request module, Begin Parallel Search module, Processing Request in Local Site, Processing Request at each Remote Site.

2.2 How do we measure

Consider that one wants to measure the time taken by a module (or a subroutine) of the DIENST server during operation. In order to do this, the start-time and the finish-time of this module has to be time-stamped. From this, the time taken by this module for a query is simply the difference.

The measurement system consists of 2 processes that perform the measurement and computation (such as means and variances of variables). A third process (DMP) manages the variable database. The measurement system captures the time spent by a query in a module and updates the performance variable for that module. The average and variance of this variable is computed for every query.

3 The demo

We have prepared a demo of a working example of the system. The demo can be found at <http://www.csi.forth.gr>.

The demo consists of three Dienst Servers running on a local area network. Our demo shows the performance of the three servers while in operation. For the demo, we show performance variables that capture the behavior of the 3 servers (D_1 , D_2 and D_3).

DIENST requests are submitted to D_1 , which performs searches, locally at D_1 , and remotely at D_2 and D_3 . We capture, via the performance variables, the following: a) the over all search response time at D_1 , b) the local search response time at D_1 , c) the remote search response time at D_2 and D_3 , d) network delays (HTTP overhead): $D_1 \leftrightarrow D_2$ and $D_1 \leftrightarrow D_3$.

We can query each of the servers for its own Performance Parameters or we can query a Dienst Server for the the Performance Parameters of all servers. We can see selected Performance Parameters and statistics graphically, using a java applet.

In order to see new values in the Graphical Performance Parameters' Display requests must be sent to the Dienst Servers. We can either make directly requests to the servers, by using them, or create artificial load (using a script).

4 Related Documents

Can be found at http://www.ics.forth.gr/Dienst/htdocs/document_menu.html, also "NCSTRL: Network Computer Science Technical Reports Library", at: <http://cs-tr.cs.cornell.edu/> "Measuring and Monitoring DIENST Behavior: A Performance Management Architecture", at: http://www.csi.forth.gr/proj/samos/html_docs/dienst.html and "Dienst Performance Measuring and Monitoring: Preliminary Results", at: http://www.csi.forth.gr/proj/samos/html_docs/results_report.html