01-CD

World Multiconference on Systemics, Cybernetics and Informatics

July 22-25, 2001
Orlando, Florida, USA

PROCEEDINGS

Volume I

Information Systems Development

Organized by IIIS
International
Institute of
Informatics
and Systemics

Member of the International
Federation of Systems Research

IFSR

Co-organized by IEEE Computer Society
(Chapter: Venezuela)

EDITORS
Nagib Callaos
Ivan Nunes da Silva
Jorge Molero

Crawling Distributed Operating System within Multiple Heterogeneous Operating Systems of Internet-connected Hosts*

Andriy SELIVONENKO, Naphtali RISHE, Toby BERK, Oksana DYGANOVA, Scott GRAHAM, Daniel MENDEZ
High Performance Database Research Center
Florida International University
Miami, FL 33199, USA

ABSTRACT

Distributed computing within the World Wide Web has been the focus of active research lately. There are various approaches to creating a heterogeneous distributed computing framework. We propose a dynamic, distributed, heterogeneous computing environment that crawls over Internet hosts. We introduce the concept of a "Compute-per-View" model, which extends the widely accepted "Pay-per-View" model. The proposed "Crawling Distributed Operating System" overcomes limitations of the "Pay-per-View" model and allows idle network bandwidth, CPU and disk resources to be used more efficiently. We demonstrate how Semantic database technology facilitates distributed databases with code replication, rollback mechanism support and the prevention of "racing condition" related problems.

Keywords: distributed operating system, Compute-per-View, Pay-per-View, Java Virtual Machine.

1. INTRODUCTION

1.1. Scope of the "Crawling Distributed Operating System" project

There are vast numbers of computing, network, and storage resources idling or not fully utilized available on the Internet [5]. Our goal is to elaborate an approach that will employ such resources and create benefits for the hosts that lend their CPU, network, and storage resources. The project "Crawling Distributed Operating System" (CDOS) facilitates collaboration of Java components over the Internet. CDOS also facilitates distributed computations, using applications that were created using native code compilers (such as C/C++ or Fortran compilers) and adjusted to specific hardware and operating systems.

1.2. Conditions and limitations of the Internet-connected host's hardware and software environment.

The principles of CDOS organization should be flexible enough to span a variety of local host conditions, since the target of CDOS is variety of Internet hosts, each of which may have different hardware, CPU instruction sets, and underlying operating system.

- CDOS should have a "non-intrusive" method of ingesting the CDOS seed into the underlying host operating system. CDOS uses the CDOS seed, which is ingested into an Internet browser's Java Virtual Machine. Therefore all machines that run Java-enabled Internet browsers can participate in the CDOS framework.
- The Java-based CDOS seed and the Java Virtual Machine core of CDOS operation overcomes security limitations, which frequently forbid native code applications that have arrived from the Internet to be invoked on the host. Security restrictions could be lifted via a security clearance that the user could give to the CDOS seed. CDOS could execute native code applications after being given native code invocation security clearance. The host browser's Java Virtual Machine settings might disallow any native code application invocation or the user might refuse to give such security clearance in some cases. CDOS could run even within such restricted conditions, since the CDOS framework is based only on Java code.
- Internet hosts are not reliable. CDOS should inherently deal with reliability issues. Furthermore, the CDOS seed could be purged from the Internet host at any time, when the user decides to close the Internet browser. CDOS should be able to deal with the sudden termination of its instance on a particular host.

2. CDOS CONCEPT

2.1. CDOS features.

The "Crawling Distributed Operating System" Java framework allows the user to:

- participate in distributed Internet computing:
  - perform compute-intensive tasks on the user's machine in background mode without imposing significant delays to foreground application services
  - render temporary storage of intermediate computing results and make them available to the Internet community
- get Internet browsing benefits as payback for performing computing tasks
- bring various parties together to participate in computing pools. (Payment will be done via direct payments and/or "barter exchange"; see Figs. 1,2):
  - parties with computing requests, who will cover expenses

* This research was supported in part by NASA (under grants NAG5-9478, NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830) and NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409).

- o parties with information services, who will get credit
- o users with computing resources, who will get credit
- o users with information needs, who will cover expenses

2.2. CDOS invocation scenario

CDOS does not need to be installed on the user computer. (The CDOS seed is shipped as an Internet Java application, which is executed while the user browses the Internet.) CDOS might be populated on a series of Web servers that are participating in CDOS-enabled computations.

We name such a set of Web servers a "CDOS pool." There could be more than one CDOS pool. CDOS is ingested into machines, which are being operated by end-users ("user hosts") from this pool/these pools. The CDOS start-up scenario consists of the following steps (see Fig. 6):

- The user comes to a site that participates in one of the CDOS pools.
- The user's browser downloads the CDOS seed as a .jar/.zip archive, which is included in an HTML page. This seed has a digital signature, which the browser uses to validate the seed's content authenticity.
- The seed initiates permission clearance dialog (see Fig. 3). The user accepts the conditions of participating in CDOS pool on one of the following levels:
  - full participation — user gives security clearance for
    - data storage on hard drives
    - connecting to various Internet hosts
    - executing native code applications

    Such users will receive maximal credit rates. Tasks that require large data exchanges or contiguous data storage will be preferentially assigned to these users.
  - full directly-connected computing participation — users gives security clearance for
    - connecting to various Internet hosts
    - executing native code applications

    Such users will receive medium credit rates. Compute and network transfer-intense tasks will be preferentially assigned to these users.
  - storage-only participation — user gives security clearance for
    - connecting to various Internet hosts
    - data storage on hard drives

    Storage-intense tasks will be preferentially assigned to these users.
  - directly-connected computing participation — user gives security clearance for
    - connecting to various Internet hosts

    Such users will receive lower credit rates. Computing tasks, whose performance does not significantly degrade when run in a Java Virtual Machine environment rather than as native code, will be preferentially assigned to these users. An example of such computations would be floating point computations with a high percentage of trigonometric functions calculations – the performance of such calculations does not degrade significantly when they are performed in Java rather than in C++.

Fig. 1. "Compute – per - View" Data Flow Model

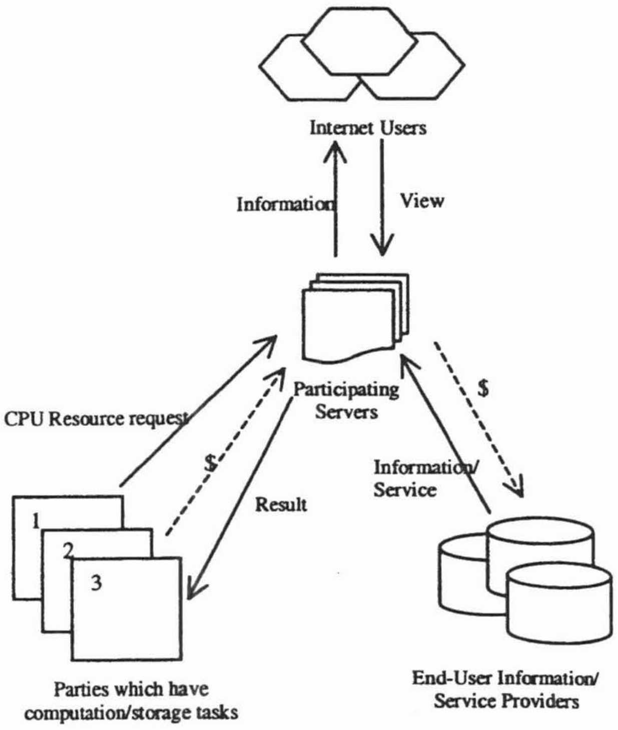
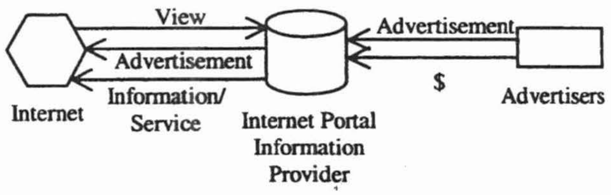


Fig. 2. Commonly accepted alternative to "Pay-per-View" Model: Advertisement-paid information services



  - limited participation — user does not give any security clearance

    Such users will receive the lowest credit rates, and will be assigned computing tasks similar to those presented in the "directly-connected computing participation" case, but with preference given to tasks where a limited number of hosts need to participate in the computations. Such users communicate with a single Leading Server, which participates in CDOS. The Leading Server for a particular user host instance of CDOS is the Web server that presented the CDOS seed into the user host's Internet browser. If a particular task requires connections to various participating hosts, such requests will be carried out via a proxy-mechanism using the Leading Server (see Fig. 4). The Proxy mechanism also enables extended horizontal and vertical communication in cases of "limited participation" in CDOS by the user host (see Fig. 5).

Most tasks could be performed even by a pool of "limited participation" user hosts. User hosts where users give at least one security clearance will perform some tasks more efficiently.

Further steps of the CDOS invocation scenario follow:

- User host starts Java CDOS seed instance.
- User host broadcasts to "leading" server host and/or to a pool of server hosts (if user initially gave "connecting to various Internet hosts" security clearance) "Available for service" message
- User browser stays on the "Leading" host web-server via an IP retaining mechanism.

2.3. CDOS functionality description.

The "leading" host web-server employs an IP retaining mechanism. This mechanism guarantees that at least one instance of the browser will stay permanently on the URL, where hostname maps to this invariable IP address. To employ an IP retaining mechanism, the Web server should emulate browsing the Internet either via an HTML frames approach or via a dynamic HTML translator/converter where remote URLs are replaced through proxy emulation.

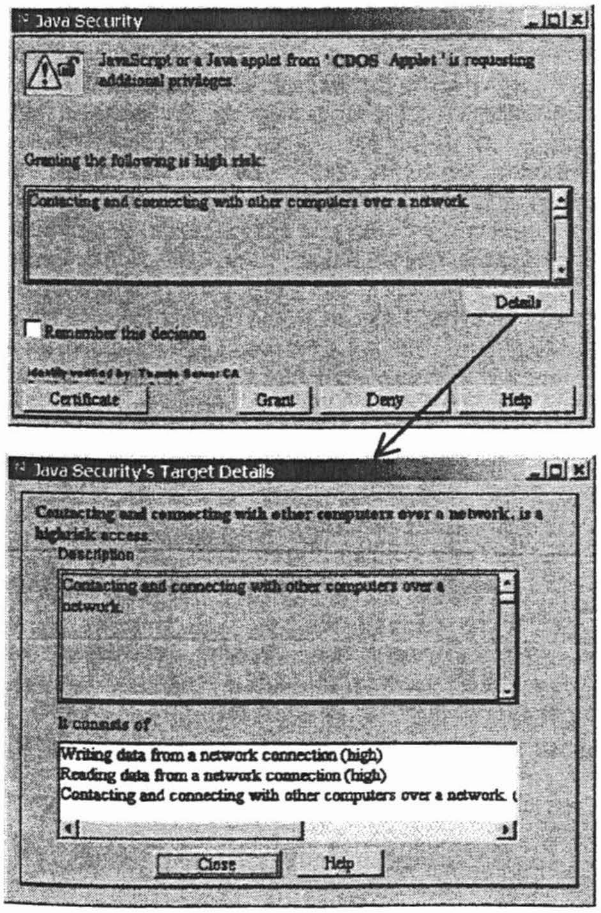


Fig. 3. Digital Signature verification process. Applet requests "contacting and connecting with other computers over a network" permission.

Java Security

JavaScript or a Java applet from 'CDOS Applet' is requesting additional privileges.

Granting the following is high risk:

Contacting and connecting with other computers over a network.

Remember this decision

Identity verified by: Thawte Server CA

Java Security's Target Details

Contacting and connecting with other computers over a network, is a highrisk access.

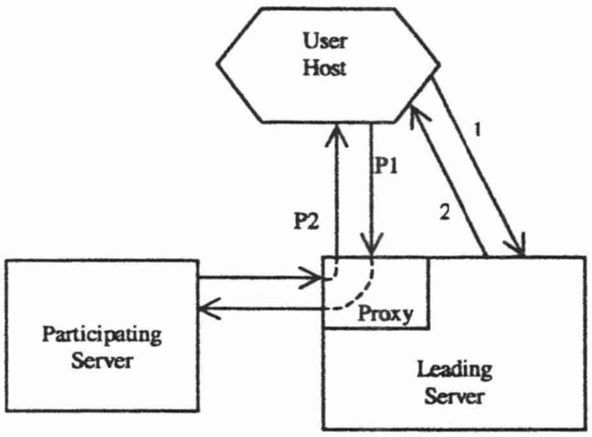
Description

Contacting and connecting with other computers over a network.

It consists of

Writing data from a network connection (high)
Reading data from a network connection (high)
Contacting and connecting with other computers over a network. (

Fig. 4. Limited participation in CDOS by User Host



1 - Direct Request from User Host to Leading Server

2 - Leading Server replies to user host directly

P1 - User Host sends request to Participating Server through Proxy service of Leading Server

P2 - Participating Server sends reply to User Host through Proxy service of Leading Server

The CDOS seed serves as the core of CDOS and supports the following CDOS primitives:

- broadcasts CDOS participating user-node availability
- specifies computing speed, network connectivity, and storage amount quota estimates
- responds to requests about CDOS participating user-node availability
- replies with current computing speed, network connectivity and storage amount quota
- registers particular services (load some algorithm implementation that will compute specific tasks)
- accepts computing/storage task via registered service
- monitors task execution
- saves rollback points for a particular task execution on designated CDOS pool server host(s) and/or other user hosts
- restores failed tasks from the latest available rollback point
- returns execution results
- monitors the status of underlying operating systems
- determines a non-intrusive current compute quota, based on current processor(s) load and user's activity
- determines storage quota, based on disk capacity, file system permission, and disk read/write load

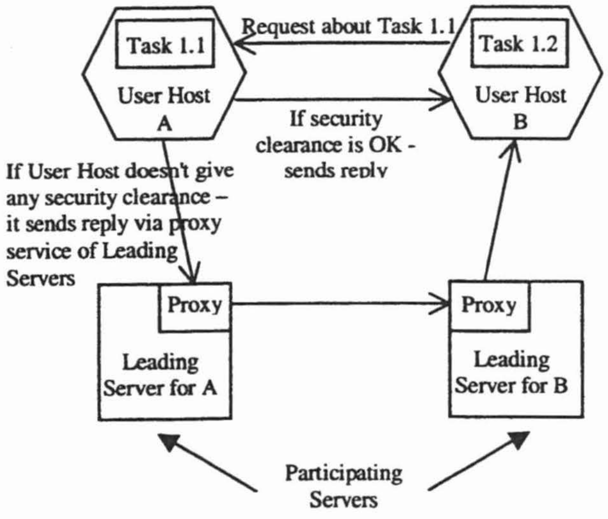
• determines network connection quota, based on current traffic, available bandwidth, and user's activity
• makes calls to underlying operating systems to carry out services that are unavailable within Java Virtual Machines, such as launching native code tasks
• CDOS addresses problems of fault tolerance.

Portion of the CDOS, which is running on host-servers, makes decisions about tasks execution on particular user-hosts and coordinates computations/storage.

3. SEMANTIC DATABASE TECHNOLOGY WITHIN CDOS

Semantic database technology serves as a back-end distributed intelligent storage subsystem. Semantic Object-Oriented Database System (Sem-ODB), developed at Florida International University's High Performance Database Center, was used within CDOS. [1, 6] Sem-ODB addresses issues of distributed databases via code replication, rollback mechanism support, and the prevention of problems related to "racing conditions."

Fig. 5. Horizontal and Vertical communication in CDOS by user hosts with and without network security clearance.


Task 1.1
User Host A
Request about Task 1.1
Task 1.2
User Host B
If security clearance is OK - sends reply
If User Host doesn't give any security clearance – it sends reply via proxy service of Leading Servers
Proxy
Leading Server for A
Proxy
Leading Server for B
Participating Servers

Within CDOS, SemODB substitutes the file semantics operations by database transactions. CDOS requires all stream input/output operations to be represented as streams, which are tunneled via HTTP protocol, on the user host level. HTTP streams could be conveyed further via proxy features of participating servers and via direct connection for the user hosts with network connectivity security clearance. Launching of the code, to be executed, is done via program code loading from the Sem-ODB. Data needed for program execution, as well as intermediary and final data are stored in Sem-ODB. Sem-ODB emulates file system semantics with database semantics. The Sem-ODB transaction mechanism supports atomic transactions, which are an efficient means of synchronization and recovery in distributed operating systems [4].

4. IMPLEMENTATION AND RESULTS

Java is a widely used choice for implementation of distributed operating systems that could be executed on various hardware platforms and host-specific operating systems [2, 3].

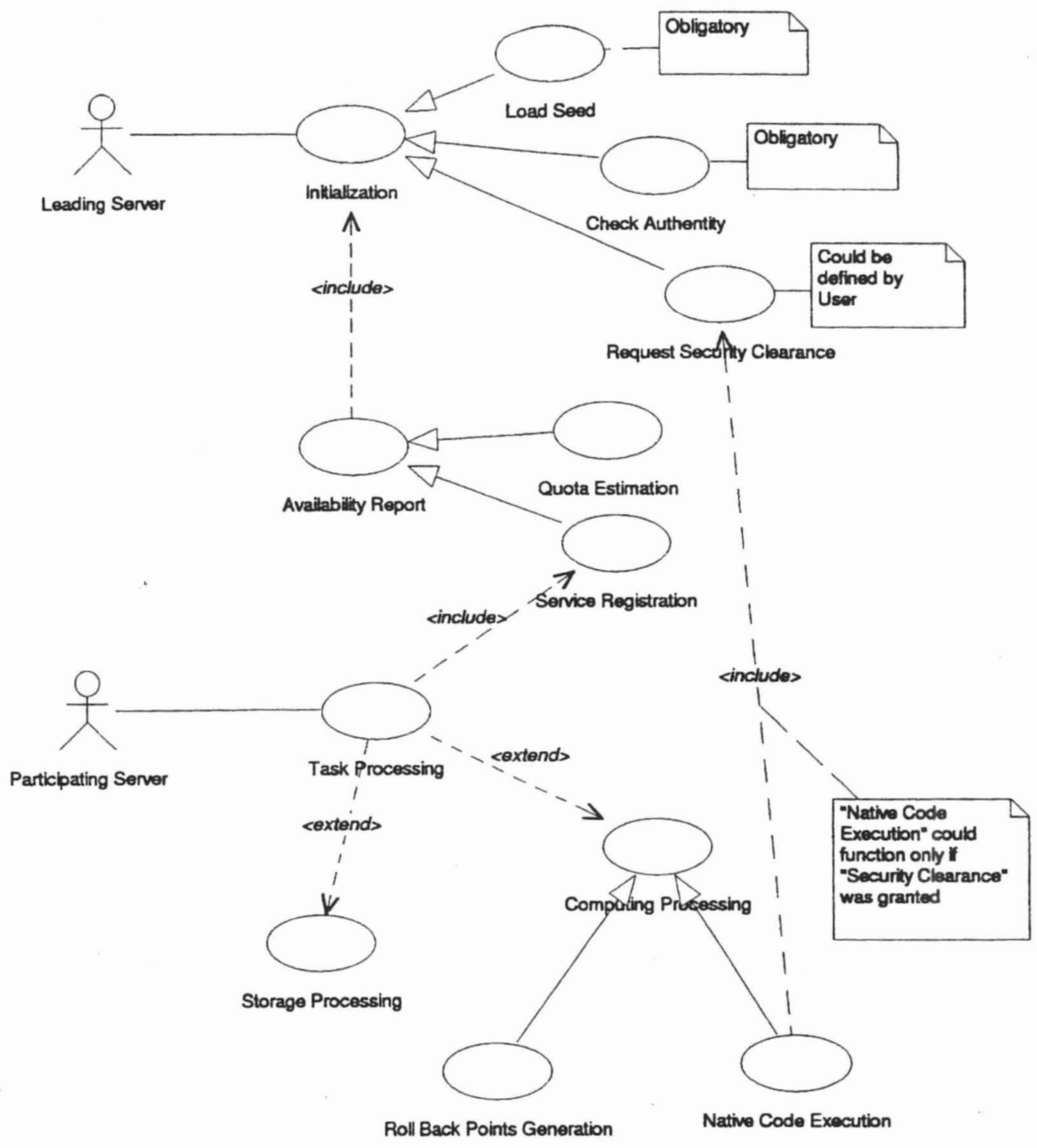
The client side of CDOS consists of the CDOS seed, implemented as a Java 1.1 compliant applet (see Fig. 6). The CDOS seed is invoked within the HTML frame, which is placed in a frameset. At the moment a user arrives to participate in a CDOS Web site, the user's browser downloads the frameset. The frameset consists of two frames: one is a hidden frame containing the CDOS seed, while other is an information frame. The information frame displays desired Web site content and keeps changing while the user browses the Internet. The hidden frame keeps the client on the IP of the Leading Server and ensures that CDOS seed is permanently loaded. Therefore, the CDOS seed is guaranteed to be resident in the client browser's Java Virtual Machine. The CDOS seed maintains a live connection to the Leading Server.

Whenever some Participating Server has a job to be computed, it propagates requests to all Participating Servers. The Participating Servers try to fetch the job from the job-initiator Participating Server's database. This job will be assigned to the first responding server or via some other selection algorithm. This job is marked in the database of the job-initiator Server as "in-process" with timestamp equal to currentTime + JobEstimatedDuration. JobEstimatedDuration is present in the database as a characteristic of this job, adjusted to the client's reported computing capability. This job is unavailable for execution in the database for the JobEstimatedDuration time since this job was sent for execution. If a response signifying job completion does not come within JobEstimatedDuration, the job-initiator sends an AYA ("Are you alive?") request to the running instance via its Leading Server. The AYA request is permitted N times as specified in the characteristics of this job in its database entry. If N is equal to 0, the job is supposed to be done within jobEstimatedDuration. If there is an answer timeout, the job is marked "available" in the database and the Client Host, which overran the time limit, is blacklisted against performing such a job. Therefore, the Client Host will not be assigned to this job again, unless there are no other non-blacklisted candidates to perform this job.

The CDOS seed loads desired Java class(es) via the Java dynamic class loading mechanism and Java Reflection API:

```
import java.lang.reflect.*;
......// exception handlers are omitted for brevity
Class loadedClass = Class.forName("NameOfTheClass") ;
Method concatMethod =
    loadedClass.getMethod("desiredMethodName",...);
concatMethod.invoke(...);
```

Fig. 6. Use Case diagram: CDOS Seed ingestion; Leading Server, Participating Server, and Client Host portion of CDOS interaction



All classes should be placed in appropriate packages to avoid name conflict. Class loading requests go to the Leading Server via http protocol. The Leading Server resolves these requests as follows:

• maps a Java package name (i.e. "package edu.fiu.cs.www") to a set of servers, which store Java classes within this package (i.e. bach.cs.fiu.edu, miami.cs.fiu.edu …) via job-initiator Server DB look-up;

• retrieves desired Java class(es) from one of the servers. If a retrieval attempt causes a timeout, the retrieval attempt is repeated with the next server in the set.

• tunnels requested Java class(es) back to Client Host via proxy mechanism.

CDOS supports the semantics of an Internet-distributed file system. There are many attempts to bring distributed files systems to the Internet level and to enrich file systems with database semantics: Sun is moving its NFS to WebNFS [7]; the Internet Engineering Task Force is creating Common Internet File System (CIFS) [8]; Oracle is merging file systems into database environments (Oracle Internet File System – iFS) [9]. CDOS uses Internet global addressing for name resolution in its file system, http protocol as its universal transport media and a database as its universal storage media. Its database mechanism supports the concurrency, security, and replication features of file systems.

At the current stage of CDOS development, it supports such file system primitives as open, read, write, and close. File streams could be opened either for read or for write. CDOS does not yet support lseek and open in read-write mode.

CDOS supports a rollback procedure that allows intermediate steps of a particular task's computations to be stored and allows the return to some rollback point, if needed. The rollback feature allows lengthy computation tasks to be split into portions that could be executed on different hosts sequentially. This feature is especially important in the CDOS environment, where the typical life span of Client Host availability is just a few hours. A particular job should implement a Java *Serializable* interface to support the CDOS rollback feature. Thus, a running Java process should serialize itself into the output stream upon a "Save Rollback" request. This stream is saved as Rollback point into the database and computations are resumed.

CDOS addresses issues of client and/or server crashes.

- If the Leading Server crashes, all CDOS seeds on clients that ingested their CDOS seed from this Leading Server cannot continue their operation. The Leading Server emulates unlimited browsing of the Internet with the browser retaining the Leading Server IP. When Leading Server crashes, the user will not be able to continue the Internet browsing and therefore will be forced to close the browser or type a new URL, which will terminate the CDOS seed. Such termination is obligatory by browser and Java Virtual Machine interoperability conventions. This makes further CDOS operation on the client that is tied to a crashed Leading Server IP impossible. If the CDOS seed were running some task at this time, this task would be terminated and removed from client memory. The Job-initiating Server will notice a timeout of the crashed Leading Server, will mark this job in the database as "available for execution" and will broadcast an invitation to execute among Participating Servers. If some rollbacks were saved before the crash, this job will be re-executed from the last rollback point.
- If CDOS crashes on the Client Host or the user closes the browser and therefore terminates CDOS before a running task is completed and its results saved into the Job-initiating Server's database, the latter will notice a timeout and re-execute this task as was demonstrated above.
- If the Job-initiating Server crashes, the Leading Server will be unable to store the results of this task or its requested rollback results. When the CDOS seed on the Client Host gets negative acknowledgement or no acknowledgement at all from the Leading Server after a rollback save attempt, it will terminate the current task.

CDOS prevents "racing condition" situations via database transactional mechanisms. On the Client Host, the CDOS seed has only one execution thread, which could load the task for execution. If the CDOS seed notices that a current task is taking more CPU/network resources than its self-imposed CDOS quota, CDOS reduces the priority of the running task.

If the Leading Server desires to execute more than one task on a particular Client Host, it gives a signal to the existing CDOS seed to spawn another instance of the CDOS seed. Multiple CDOS seeds don't communicate between themselves on the Client Host side to reduce CDOS seed code size and complexity. The Leading Server makes decisions about CDOS seeds spawning and termination.

5. CONCLUSION

We presented a "Crawled Distributed Operating System." It facilitates flexible, lightweight Internet computational media, which works over heterogeneous hardware and software environments without requiring any explicit running code installations and administration on the user side. The proposed CDOS was successfully implemented and the system is scalable and platform independent. An alternative business model "Compute-per-View" was introduced. We demonstrated that the CDOS approach makes such model existence possible. We showed that "Compute-per-View" extends the traditional "Pay-per-View" model as well as a conventional advertisement-based Internet content technology. CDOS bridges the gap between the file system semantics, execution control semantics, and database semantics. We anticipate that CDOS model might be applicable to a wide variety of computational tasks such as statistical calculations, thermodynamic/hydrodynamic/ nuclear simulations, 3D rendering, financial analysis, and scientific experiments.

6. REFERENCES

[1] N. Rishe, *Database Design: The Semantic Approach,* McGraw-Hill,Inc. N.Y., N.Y, 1992.

[2] Tommy Thorn, *Programming languages for mobile code,* ACM Comput. Surv. 29, 3 (Sep. 1997), Pages 213 - 239

[3] Michael Swaine, *Programming paradigms*, Dr. Dobbs J. Nov. 1998, Article 11

[4] Thomas W. Page, Matthew J. Weinstein and Gerald J. Popek, *Genesis: a distributed database operating system,* Proceedings of the 1985 international conference on Management of data, 1985, Pages 374 - 387

[5] Neil Randall, *Power to the People,* PC Magazine, April 24, 2001, Pages 80-84

[6] N. Rishe, W. Sun, D. Barton, Y.Deng, C. Orji, M. Alexopoulus, L., Loureiro, C. Ordonez, M. Sanchez, A. Shaposhnikov, "Florida International University High Performance Database Research Center", SIGMOD Record, 1995, vol.24, no. 3, pp.71-76.

[7] http://www.sun.com/software/webnfs/

[8] http://msdn.microsoft.com/workshop/networking/cifs

[9] http://otn.oracle.com/products/ifs/pdf/ifsfo1_1.pdf

980-07-7541-2
57000>
9 789800 775417
ISBN: 980-07-7541-2