Automation Will Solve Most of Our Problems

CS553: Internet Services, Spring 2006

Abstract
Availability of an Internet service is negatively impacted due to operator errors and software bugs. In today's Internet services operator errors are one of the sources of downtime that is further exacerbated by the software bugs. Software bugs affect continuous operation of a computer system that requires human operator intervention for correction. Similarly, operator errors are one of the sources of service latency and downtime affecting the availability of an Internet service.

In this paper we take the position that one can achieve a high degree of availability by providing the appropriate automation tools for
  i) updating deployed software for bug fixes and
  ii) operators to install, configure, integrate and fine tune the Internet service.

1. Introduction
Software bugs are going to exist as long as software is being continuously enhanced and modified to incorporate new functionality demanded by the end users. Operator errors exist partly due to lack of automation tools available to assist the operator. With these static facts in mind, we argue that through automated timely updates of the software base for detected faults using suggested hierarchical architecture and automated tools to assist human operator can increase the service availability to four nines or higher.

The study carried out by Jim Gray [1] for fault-tolerant computer systems indicates that software bugs followed by operator errors are the largest contributor to service outage in 1989. Another study carried out lately by David Oppenheimer [2] in 2003 for large-scale Internet services also clearly showed results where operator error and software component failures were the major source of failure.

Table 1 below shows the failure percentages obtained from these studies. As we can see that the totals of these two errors is significant and if one eliminates these errors though automation then one can obtain availability of up to four nines or more. Online service availability is calculated as follows:

\[
\text{Uptime} \\
\text{Service Availability (nines)} = \frac{\text{Uptime}}{(\text{Downtime} + \text{Uptime})}
\]

Four nines allows for a downtime of up to 51 minutes per year for a service. This is easily achievable as most of the failures are due to operator errors and software faults
as shown by the Totals column of Table 1. The hardware quality has improved and there is high level of fault tolerance available at the hardware level.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tandem Systems[1]</td>
<td>16%</td>
<td>62%</td>
<td></td>
<td></td>
<td>78%</td>
</tr>
<tr>
<td>Online Service [2]</td>
<td></td>
<td></td>
<td>33%</td>
<td>27%</td>
<td>60%</td>
</tr>
<tr>
<td>Content Service [2]</td>
<td></td>
<td></td>
<td>36%</td>
<td>25%</td>
<td>61%</td>
</tr>
<tr>
<td>Read Mostly Service[2]</td>
<td></td>
<td></td>
<td>19%</td>
<td>24%</td>
<td>53%</td>
</tr>
</tbody>
</table>

Table 1: Operator Errors and Software Faults: Op column shows % of failures due to Operator Errors, SW column shows % of failures due to Software Errors. [1] is study carried out by Jim Gray and [2] is study carried out by David Oppenheimer.

Source of of both software bugs and operator errors are human elements or characteristics, following section briefly covers the human limitations that manifest in the form of bugs and errors. In section 3 we cover the counterclaim to our argument, in section 4 we cover the Software bugs and how to automate the deployment of bug fixes. In Section 5 we discuss automation to assist the operator to mitigate the operator related errors.

2. Human Limitations
People make errors, tend to forget, become impatient, get confused, and tend to be lazy. These very limitations manifest in the software produced in the form of bugs. People also have limitations in the form of sensory, cognitive and motor overload. In case of operations these are manifested when they tend to miss the opportunity to correct the known error in time, or inadvertently kill a process for example.

3. Counterclaims
Automation will not solve most of our problems is the counterclaim, it is hard to automate the task of bug fixing that is required for correct behavior of software used in a service. This is not true because of the advancement in the security and Internet technology, it is possible to automate this task, by having appropriate secure channels for updating the software bugs automatically over the Internet.

The other counterclaim is that it is impossible to provide automation tools to assist an operator, because they are difficult to build and are too specific to an application. This not true because now a days the OS install is totally seamless on new PCs, just power on for the first time and installation begins with minimal user input.
Similarly, special software packages are available from the vendors for uniform installation of OS on a set of computers on a LAN or WAN, example of this is Jump Start for Sun Solaris OS [6]. The automated install packages are provided by the vendors. For large distributed database systems like DNS used for name resolution on Internet, there are automated tools available to check the system configuration[5].

4. Software Bugs
A software bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working as intended, or produces an incorrect result. When an Internet service software has a bug, it impacts its availability when affected piece of code executes. For correction, one needs to debug and update the software used in the Internet services. Jim Gray [1] has cited that Software faults where responsible for over 60% of the times for system outages. In this paper we believe that most of the faults occurring are due to known bugs that have a bug fix available that needs to be deployed. This is because in critical production systems only software that is well tested is used. In the following section we describe an architecture that would allow for automated update of software bug fixes.

4.1 Architecture for Automated Update of Software Bug Fixes
Software used in production system used for busy Internet service is well tested both for functionality and stress. Even after these tests, bugs do manifested for the simple reason that there arises a unique situation that was not tested before. In the present day, software bugs can be masked by use of clustered service, and partitioning the node that has generated the fault.

![Figure 1. Architecture for automated software update](image-url)
Here we describe an architecture by using which we believe one can minimize the software bugs in Internet services. Service supervisor modules can monitor the worker service as shown in figure 1. Service supervisor assigns tasks to the worker service that in turn fulfills the request or generates a fault. The purpose of the supervisory service is to check for any new patches or releases of the software and accordingly update the worker service software. The supervisory service can check for the new updates on the vendor sites. In case of any faults generated by the worker service, the service supervisor can redirect the request to a different worker service. Supervisor service can then check for of any possible bug fix matching the type of fault generated, if the patch is available then it can immediately update the software for worker service.

This way service software can be kept up-to-date automatically which results in higher availability of the overall system. Given that software bugs are the major source of downtimes, the described architecture can be used where software bugs are fixed before their effect is noticed by the end-users. This architecture allows for automatic update of software for bug fixes as soon as they are released by the vendor.

5. Operator Errors
In the study carried out by Jim Gray [1] we note that operator errors were the second largest cause for system outages. Another study carried out twelve years latter by David Oppenheimer [2] for Internet Services shows that operator errors are the prime cause of system failure and degraded performance (in case of two of the services studied). Table 1 above shows these errors by system type and year. Experiments carried out with operators with various levels of skills by Richard Martin [3] also points out that operator errors are the prime suspect in the service downtime.

Reason why operator errors have topped in recent studies is that, configuring and operating large scale Internet services is complex and lack automation tools to assist the operator. Just to give an example, setting up a user for FTP access requires changes at various service points like NIS+ directory, firewall access, and file server space assignment. These type of tasks are generally carried out without any automated tools and the process is error prone.

Operator errors are caused by the human operators who are possibly setting up the system initially, or are fixing a reported problem, or are fine tuning a system for higher loads, or it could be during their routine task of housekeeping a file system. These errors happen inadvertently by the operators, and the main reason being that they do not have appropriate automated tools to test the correctness of changes or configurations they have made. Another reason operator errors is that the system to be configured involves many steps and has high configuration complexity. Here we argue that by providing automation tools to operators one can achieve a high degree of availability.
Operator errors can be broadly divided into two categories:

i) Configurations: Many programs depend on parametric settings that define the behavior of the system. These parameters need to be set correctly depending on requirements. On startup the program reads these parameters for initial setup. These parameters or variables are generally stored in a file and require manual setting using an editor in case of Apache web server. In setting up the parameters, an operator can miss a parameter or create a syntax error in the file that affects the performance of the program being configured. Service configuration error is the largest of the operator errors[2].

ii) Procedures: Operators carry out certain tasks from time to time for upkeep of the system, like upgrading software, taking backups, cleaning up log file system, recreating index files for compaction. Operator tend to make mistakes when procedures have many steps.

Automation tools can be used to check for any configuration anomalies after initial configuration and also the procedures with many steps can be automated to minimize the errors. In the following section we give an example of a widely used Internet service that is prone to configuration errors.

### 5.1 DNS Configuration Errors

In a study conducted by Vasileios Pappas [4] on the DNS systems reveals that service availability is seriously affected by configuration errors. DNS system provides robust and scalable name-to-address mapping. DNS is widely used for name resolution on the Internet.

<table>
<thead>
<tr>
<th>Configuration Error Type</th>
<th>% of DNS Zones Affected in the Sample Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lame Delegation</td>
<td>15%</td>
</tr>
<tr>
<td>Delegation Inconsistency</td>
<td>21%</td>
</tr>
<tr>
<td>Diminished Server Redundancy</td>
<td>45%</td>
</tr>
<tr>
<td>Cyclic Dependency</td>
<td>2%</td>
</tr>
</tbody>
</table>

Table 2: DNS Configuration Errors

Table 2 above shows the prevalence of configuration errors detected, in a sample of DNS zones, that affect the availability of the DNS service. Lame delegation is a type of configuration error where by the DNS operator or administrator delegates the name
resolution of a domain name to a zone that does not keep the record for the delegated
name, as a result of this queries sent to the delegated zone results in error. Delegation
inconsistency occurs when the parent zone points to a set of name servers for the child
zone that is different from the set stored at the child zone. Diminished server redundancy
is mainly caused by placement of all the authoritative name servers in the same /24
address prefix, implying that failure to reach the /24 network results in name service
failure (Microsoft’s authoritative servers were unavailable due to this type of
misconfiguration in 2001 [4]). In case of cyclic dependency, zone X depends on zone Y
and zone Y in turn depends on zone X for information.

Automation tool can be developed as discussed in [5] to detect these type of
configuration errors. This tool detects configuration errors by identifying violations of a
correct configuration, provides monitoring from multiple points, and gives a graphical
user interface for operators to check for errors.

What we have demonstrated here is that by provision of automation tools to detect
the errors, will eliminated the configuration related errors and improve the availability.

6. Conclusion
Automation tools to assist the operator and automated upgrade of software used will
increase the availability to four nines because most of the service failure and degradation
in today's Internet service are originating due to operator errors and software bugs. A
systematic approach is required to build the automation tools, whereby one has to first
carry out a study of error types a service is suffering from and then build the tool to
specifically attack the error types. Software errors can be eliminated by using most up-
to-date bug fixed software for a service, and this upgrading process can be automated. If
these two areas of operator error and software bugs is covered then availability will
increase to four nines or above.

References:
   Reliability.
2. David Oppenheimer, Archana Ganapathi, and David A. Patterson. Why Do Internet Services Fail, and
   and Dealing with Operator Mistakes in Internet Services. Proceedings of the 6th Symposium on
4. Vasileios Pappas, Zhiguo Xu, Songwu Lu, Daniel Massey, Andreas Terzis, Lixia Zhang. Impact of
   Configuration Errors on DNS Robustness. SIGCOMM 2004.
   SIGCOMM 2004.