the Availability Digest

Let's Get an Availability Benchmark

June 2007

Benchmarks have been around almost as long as computers, Benchmarks are immensely valuable to buyers of data processing systems for comparing cost and performance of the systems which they are considering. They are of equal value to system vendors to compare the competiveness of their products with others.

In the early days of computing, there were no formal benchmarking specifications; and vendors were free to define their own benchmarks. This, of course, led to a great disparity in results and benchmark claims. In response to this problem, several formal and audited benchmarks have evolved, such as those from the Standards and Performance Evaluation Corporation (SPEC)[1] and the Transaction Processing Performance Council (TPC).[2]

Virtually all of today's accepted benchmarks focus on performance and cost. But what good is a super fast system if it is down and not available to its users? It seems that system availability is just as important today to system purchasers as is raw performance. Shouldn't availability be part of these benchmarks?

Performance Benchmarks

As mentioned above, primary examples of today's benchmark specifications are those from SPEC and TPC. The SPEC benchmarks focus on raw processing power and are useful for comparing system performance for applications such as graphics, mail servers, network file systems, Web servers, and high performance computing (HPC) applications.

The TPC benchmarks, however, deal with transaction processing systems. As such, they relate more closely to the topics of interest to us here at the Availability Digest.

Transaction Processing Benchmarketing

The Transaction Processing Performance Council was founded in 1988 by Omni Serlin and Tom Sawyer in response to the growing problem of "benchmarketing," the inappropriate use of questionable benchmark results in marketing promotions and competitive comparisons. At the time, the accepted benchmarks were the TP1 benchmark and the Debit/Credit benchmark (newly proposed by Jim Gray of Tandem Computers).

TPC's first benchmark was TPC-A, which was a formalization of the TP1 and Debit/Credit benchmarks. However, even though a formal and accepted benchmark for system performance now existed, there continued to be many complaints of benchmarketing.

[1] www.spec.org
[2] www.tpc.org

In response, TPC initiated a review process wherein each benchmark test had to be extensively documented and then audited by TPC before it could be published as a formal TPC benchmark result.

Today, all published results of TPC benchmarks have been audited and verified by TPC.

Current TPC Benchmarks

The TPC benchmarks have evolved significantly over the years and now include:

- TPC-C, which is an extended version of TPC-A. It deals with the performance of transaction processing systems, including networking, processing, and database access. It specifies a fairly extensive mix of transactions in an order management environment.

- TPC-H, which measures decision support for ad-hoc queries.

- TPC-App, which focuses on Web Application Servers.

- TPC-E, which is a TPC-C-like benchmark but which includes many requirements to make it more closely resemble the enterprise computing infrastructure. TPC-E is still in the TPC approval process.

The TPC-C benchmark is the one most pertinent to this discussion.

TPC Performance Metrics

The results of the TPC-C transaction benchmark are provided as two numbers, along with a detailed description of the configuration of the system that achieved the results. These measures are:

- *tpmC*, the number of TPC-C transactions per minute that the system was able to consistently process.
- *Price/tpmC* - the cost of the system expressed as its cost for one transaction per minute (tpm) of capacity.

For instance, if a system is able to process 100 transactions per minute and costs $100,000, its cost metric is $1,000 per tpm.

System performance has come a long way. The first TPC benchmarks showed a capacity in the order of 2,000 transactions per minute (using a TPC-A transaction, simpler than today's TPC-C transaction) and a cost of about $400 per tpm. Today, commercially available systems are achieving 4,000,000 TPC-C transactions per minute and cost less than $3 per tpm. This is roughly a 2,000-fold increase in performance and a 130-fold reduction in price in less than 20 years, equating to over a 250,000-fold increase in price-performance!

Availability Benchmarks

Downtime Costs

In the early days of benchmarking, performance was king. There was not much attention given to system reliability because (a) high-availability technology was not there yet and (b) systems did not have to run 24x7.

But today, as markets become global and as the Web becomes ubiquitous, systems are expected to operate continually. Not only is downtime unacceptable, but its cost can be enormous. A USA Today survey of 200 data center managers found that over 80% of these managers reported that their downtime costs exceeded $50,000 per hour. For over 25%, downtime cost exceeded $500,000 per hour.[3]

As a result, today's cost of system ownership can be greatly influenced by downtime. Think about it. Consider a critical application in which the cost of downtime is $100,000 per hour. The company running the application chooses a system which costs $1 million dollars and which has an availability of three 9s (considered today to be a "high availability" system). An availability of three 9s means that the system will be expected to be down an average of only eight hours per year. However, over the five-year life of this million dollar system, it will be down about 40 hours at a cost of $4 million dollars.

$1 million dollars for the system and $4 million dollars for downtime. There has to be a better way.

One solution to this problem is to provide purchasers of systems not only performance/cost metrics but also availability measures. A company could then make a much more informed decision as to the system which would be best for its purposes.

For instance, consider two systems, both of which meet a company's performance requirements. System A costs $200,000 and has an availability of three 9s (eight hours of downtime per year). System B costs $1 million dollars and has an availability of four 9s (0.8 hours of downtime per year). Over a five-year system life, System A can expect to be down for forty hours and System B can be expected to be down for four hours.

If downtime costs the company $10,000 per hour, the initial cost plus the cost of downtime for System A is $600,000. That same cost for System B is $1,040,000. System A is clearly more attractive.

However, if the company's cost of downtime is $100,000 per hour, the same costs for System A are $4,200,000 and those costs for System B are $1,400,000. System B is clearly the winner in this case.

One interesting observation casts some doubt on this analysis. Too many times, a manager is judged by his short-term performance. Consequently, he may choose System A without regard to its long term cost only because it has a smaller initial cost. He is now a hero and leaves the long-term damage to his successor.

Availability Metrics

How do we characterize availability? The classic way is to provide the probability that the system will be up, or operational (or alternatively that it will be down, which is 1 – the probability that it will be up).

For highly available systems, this probability can be awkward to express. For instance, it is difficult to say or read .999999. Does this number have five 9s in it? Six 9s? Seven 9s? Therefore, it is conventional to express availability as a number of nines. The above availability would be expressed as six 9s. A system with three 9s availability would be up .999 of the time (99.9%) and would be down 1 - .999 = .001 of the time (0.1%).

[3] USA Today, page 1B; July 3, 2006.

There are many other metrics that can be used. Common metrics include the *mean time between failures (MTBF)*, which is the average time that a system can be expected to be up, and the *mean time to repair (MTR),* which is the average time that it will take to return the system to service once it has failed.

Availability and failure probability are related to MTBF and MTR as follows:

$$\text{availability} = A = \frac{MTBF}{MTBF + MTR}$$

$$\text{probability of failure} = F = 1 - A \approx \frac{MTR}{MTBF}$$

Measuring Availability

Before measuring availability, it must be more carefully defined. Most data processing professionals today accept that availability is not simply the proportion of time that a system is up. Availability has to be measured from the user's perspective. If a network faults denies access to 10% of the users, the system is up but not so far as those users are concerned. If the system is expected to provide subsecond response times but is currently responding in 10 seconds, it may well be considered to be useless and therefore be down to the users.

One form for the definition of availability might be that the system is up if it is serving 60% of the users and is responding to those users within the response time requirement 90% of the time. Another, perhaps more useful, method would be to use partial availabilities in the availability calculation. For instance, following the above example, it the system were satisfactorily serving 60% of its users, then during that time it has an availability of 60%.

One problem with considering an availability benchmark is that availability can be very hard to measure and verify. For instance, consider a system with an availability of four 9s and thus giving an expected downtime of 0.8 hours per year. This does not mean that the system will be down 48 minutes every year. It is more likely that it will fail occasionally and will take longer to return to service. For instance, experience may show that this system in the field fails about once every five years and requires an average of four hours to return to service.

We cannot reasonably measure the availability of these systems for benchmarking purposes. Even if there were many of these systems in the field, by the time we accumulated reliable data, the system would probably be obsolete and no longer sold. Looking over our shoulders at availability provides no useful service to potential buyers.

However, today this is all we have but in a more generic sense. There have been many studies of the availabilities of various systems based on field experience. For instance, in 2002, Gartner Group published based on field statistics the following availabilities for various systems:

HP NonStop	.9999
Mainframe	.999
Open VMS	.998
AS400	.998
HPUX	.996
Tru64	.996
Solaris	.995
NT Cluster	.992 - .995

These availabilities took into account all sources of failure, such as those caused by hardware, vendor software, application software, operator error, networks, and environmental faults (power,

air conditioning, etc.). Though these results are quite dated, they have been roughly verified by several later studies.

This is the best we have today for availability benchmarks. They are very broad-brush, they are outdated, and they lack formal verification.

So What Do We Do

Availability benchmarking is too important today to simply give up and say that it isn't possible. Dean Brock of Data General has submitted an excellent TPC white paper that deals with this subject.[4] An approach that he suggests evaluates the system architecture and gives a score for various availability attributes. These attributes could include:

- **System Architecture**
  To what extent can the system recover from a processing node failure:
  - Single node
  - Cluster
  - Active/active
  - Remote data mirroring
  - Demonstrated failover or recovery time

- **No Single Point of Failure within a Single Node**
  Every critical component should have a backup which is put into service automatically upon the failure of its mate. These components include:
  - Processors
  - Memory
  - Controllers
  - Controller paths
  - Disks (mirrored gets a higher score than RAID 5)
  - Networks
  - Power supplies
  - Fans

- **Hot Repair of System Components**
  Failed components can be removed and replaced with operational components without taking down the system:
  - All of the components listed above

- **Disaster Backup and Recovery**
  - Availability of the backup site
  - Currency of backup data at the backup site:
    - Backup tapes
    - Virtual tape
    - Asynchronous replication
    - Synchronous replication
  - Demonstrated time to put the backup site into service

- **System Administration Facilities**
  - Monitoring
  - Fault reporting
  - Automatic recovery

[4] http://www.tpc.org/information/other/articles/ha.asp

- **Environmental Needs**
  - Size of backup UPS power source included in the configuration
  - Size of backup air conditioning included in the configuration

- **Vendor QA Processes**
  - Software QA
  - Hardware QA

This is undoubtedly only a partial list and will certainly grow if this approach becomes accepted. Rather than simply a checklist of attributes, a careful availability analysis could assign weights to these attributes. Based on these weights, an availability score could then be determined; and systems could be compared relative to their expected availability (one will never know what the real availability of a system is until long after it is purchased).

A complexity in the weighting of attributes is that many weights will be conditional. For instance, the impact of nodal single points of failure is much more important in single-node configurations than it is in clusters or active/active systems.

One problem not addressed in the above discussion is how to relate an availability score to system downtime so that the costs of downtime can be estimated. It may be that some years of field experience can yield a relation to the availability score and the probability of downtime.

Summary

We have spent a decade perfecting performance measuring via formal benchmarks. It is now time to start to consider how we can add availability measures to those results.

There are many problems to solve in order to achieve meaningful availability benchmarks, primarily due to the infrequency of failures in the field. However, a careful analysis of availability attributes can lead to a good start toward obtaining useful availability metrics.