Continuous Improvement
And the
Ban Bottlenecks® Methodology

A White Paper from:

Transaction Design, Inc.
1-415-256-8369
inform@banbottlenecks.com
www.banbottlenecks.com

Table of Contents:

Overview:

Computer systems management is a daunting challenge. In the on-line transaction processing (OLTP) arena, a system manager must deal with the reality of constantly changing demand on the system, changing system and application software, changing network environment, changing staff, and occasionally a changing management or corporate structure. This much change, unless it's properly managed, causes "continuous degradation" of the computing environment

In the face of all this change, how can you maintain your discipline, reduce risk, and implement "continuous improvement?"

The answer is Ban Bottlenecks®. This service by Transaction Design, Inc. (TDI) performs a methodology which assists the system manager in monitoring and managing the factors affecting the system. This methodology embodies the "best practice" of computer systems performance management. The results of this methodology produce the opportunity for continuous improvement of the entire OLTP environment when management is focused on providing efficient highly-available computing.

Inviting Disaster:

> *We know from the long history of system failures that most problems have been like Apollo 13 in that they offered clues before the full-fledged emergency. "When has an accident occurred which has not had a precursor incident?" asks C.O. Miller, retired chief of NTSB's Aviation Safety Bureau. The answer, he says, is basically never.*
>
> Inviting Disaster: Lessons From the Edge of Technology
> (James Chiles, HarperCollins, 2001)

System Management Practices:

Management by Exception:

System managers have been trained to manage a system by exception. That is, they pay attention to the system when something draws their attention. This is the "squeaky wheel" theory of management. Usually this is represented by a set of rules or alert conditions implemented on a system which causes a flag to be raised when one of the alert conditions is met. In effect, the system gets attention when there is a problem or about to be a problem.

Some sites do this well, and some sites do this poorly. A lot of thought has to go into the design of the alarm rules. If too lax, the alarm goes off only when a problem has occurred, and that is too late. If too tight, the alarm goes off too frequently and ends up being ignored and therefore worthless. All too frequently alarms are handled by a separate operations group, and the system administration or applications group may not hear about or track the alarms.

Alarms aren't a sufficient way to manage the system. We like to say that "Capacity management is not a video game!" Unless someone is tracking the application workload and projecting it into the next peak season, the site is very vulnerable to an unplanned outage.

Performance/Capacity Management Practice:

Instead of “management by exception,” managers must employ an approach of *proactive* capacity management. However, true capacity management is rarely performed in an organization. If it is performed at all, it usually has too narrow a focus to be truly effective in the mission-critical OLTP environment. All too often midrange performance analysis is delegated to the mainframe capacity group, which doesn't understand issues which are unique to the midrange computer or application architecture. Often we see that the mainframe group concern themselves with tracking CPU and disk usage and little else.

Post-Mortem:

> *"Sure we have capacity planning meetings. But all we discuss are post-mortems."*

We know sites where the midrange systems administration group is responsible for performance and capacity management. There are several realities that make this arrangement less than optimal:

- System administration teams are frequently stretched thin, handling multiple architectures and larger numbers of systems.
- Performance and capacity data collection is a time-consuming and tedious job.
- There is so much data that must be collected and analyzed, covering CPUs, memory, processes, channels, disk busy, disk space, communications links, and the application, that the job is daunting if not impractical without the proper tools and methodology.
- Capacity planning is often abandoned, banished to a dark corner, and kept dormant until a critical problem makes everyone recall that they once tried to be proactive about this stuff.

Sizing The System:

> *"Sizing a system for the average demand is like building a bridge for the average size of a sailboat's mast. It guarantees problems"*

How Big?

The reality is that the system must be sized to handle the *upcoming* peak workload. Every application experiences cycles. There are peak times each day, peak days each week and each month, and usually a peak day or peak season each year. ATM systems have to deal with deposits on paydays and withdrawals on weekends, especially holidays like Easter and Mother's Day. POS systems always have to deal with the Christmas season. Pharmacy systems have to deal with flu season. Gas stations deal with the summer vacation drive time. Brokerages and exchanges have to deal with wild market swings.

We've had planners tell us "The system's fine. It's only running about 30% busy." We also know sites where the seasonal peak is 3 to 5 times the average workload. If an administrator doesn't know what the upcoming peak is going to throw at the system he or she isn't doing their job.

Growth, Planned and Unplanned:

If the demand on a system is flat or declining, there may be no need to talk. But our experience is that most systems are experiencing growth in demand, either from normal business growth, or from the results of mergers or acquisitions. This growth must be tracked and projected.

There is another source of unexpected demand. Marketing and new product campaigns can be unexpectedly successful. We once saw a customer introduce a new type of gift card in their business.

This card was wildly successful, with the result that the after-Christmas traffic was 8 times the "usual" traffic. No one told the systems department, and the system failed.

"But I Only Changed... And It Shouldn't Have Affected...":

Software, too, is a moving target. In most shops there are constant bug fixes and enhancements to the application software suite. Sometimes, changes have unintended consequences. We've seen:

- Fixes that created new problems;
- Enhancements that increased the CPU or disk I/O cost of a transaction;
- System software upgrades that created unexpected locks or bottlenecks.

Continuous Improvement Through Ban Bottlenecks:

Management consultants often talk about 'continuous improvement' programs. The goal of these initiatives is to accomplish major leaps in systemic performance – be it speed, throughput, stability or customer satisfaction – one small step at a time. You can't pull that off without a measuring stick or a methodology. Ban Bottlenecks gives you both of these.

Ban Bottlenecks Is Not:

Ban Bottlenecks is not real time. A site still needs its alarm systems to notify operations and systems if something is going wrong.

Ban Bottlenecks Is:

Ban Bottlenecks is the "best practice" for ongoing performance management. It integrates data collection and presentation with filtering, peak detection, trend analysis, and problem detection. It is your basis for continuous performance. With Ban Bottlenecks the system manager can see where the system has been, and by projection, where it has to go.

Avenues of Improvement:

Ban Bottlenecks provides you with a measuring stick and methodology to gain continuous improvement from: your application; system components such as CPU, processes, and disks; and from your network.

Application:

Ban Bottlenecks always reports on application traffic. System capacity numbers are meaningless unless they can be related to the application workload. We always work with the client to mutually define a "transaction." In banking, a transaction may be a message-based ATM or POS financial transaction. Or a transaction may be a user accessing the system and retrieving a cash-management report. In a pharmacy, a transaction may be an insurance claim coupled with a drug-interaction query. In a gasoline retailer, a transaction may be a pre-authorization at the gas pump coupled with a completion message. In a data switch, a transaction may be a batch transmission or reception.

On the simplest level Ban Bottlenecks will count system-level messages going into an application, and attempt to relate that to the "transaction." Ideally, however, Ban Bottlenecks will extract from the application a wealth of information about the transaction stream. For example:

ATM/POS:
Retailer POS:
Pharmacy: Transactions and response time by originator and by authorizer; Transactions by type; Transactions by response code and by reversal code. We look for the peak transaction rate, the worst response time, the highest reversal rate. (OpeN/2, ON/2, HN/2, BASE24, RxView)

Stock Market Ticker plant:
: Quotes received by the ticker plant.

Stock Market Distribution:
: Quote and trade info sent to the markets and clients.

Stock Market Trading system:
: Trades & contracts; Quotes generated and distributed.

Banking Cash Management:
: Peak report retrieval time; Requests by service type; Simultaneous users on the system; Simultaneous users within a service; Average session length; Peak and trend analysis. (Forte)

Warehouse Management:
: Number of picks, puts, and obos (outbound orders). OBO status (where an order is in the shipping cycle). (Logisticon WMS)

Data Communications:
: Collections and transmissions (jobs) by originator and receiver; Job size in records; Job duration; Job completion status; Communication device traffic analysis. (DataExpress, Network Express)

WWW server:
: Requests; Pages and bytes set. Unique clients.

Continuous application improvement comes from our ability to filter the transaction log data and search for periods of poor response time, high reject rates, and peak demand. We investigate the problems and recommend avoidance procedures. We trend and project the peaks, and look for processing bottlenecks. We can frequently uncover problems related to peer or back-end processing issues.

Effectiveness:

Continuous application improvement also comes from our ability to report on the effectiveness of the application. We track changes to the cost of a transaction over time. We report on the cost of a transaction in terms of CPU usage, disk I/O, memory, etc. per transaction. From this we can determine if there has been a change in the efficiency of the application, or if background processing is interfering with online processing.

CPU:

Continuous improvement comes from our ability to discover processing problems. A minute is an eternity when you're processing a high rate of transactions per second. We filter and chart the CPU data we capture, looking for periods of heavy usage. We report CPU usage as an average *and* as a distribution, since distributions highlight bursty behavior and averages do not.

For symmetrical multiprocessors, we can detect if a process is hogging a CPU. For asymmetrical multiprocessors, we can see if the workload is properly balanced across processors. We concentrate on periods of peak demand.

At the same time as we are analyzing the CPU data, we are also filtering it to find problems with page faults, available memory, queuing, interrupt processing, and channel activity.

Processes:

Continuous improvement comes from capturing and analyzing the data from every process on the system during times of peak demand. We continually collect process data, so that it is available for problem and trend research. Each week we create a report of the processes on the system during the peak half-hour of demand. This report is then uploaded for further analysis.

As a result or our Ban Bottlenecks methodology, we can find:

- Processes that are too busy;
- Processes that are starved for CPU or needing a boost in priority;
- Processes that have memory leaks;
- Processes that have poor memory management design;
- Processes that have queue backlogs;
- Unexpected transient processes;
- Opportunities for re-scheduling background processing;

Disks:

Continuous improvement comes from capturing, filtering, and charting the data from every disk on the system. Any individual disk can quickly become a bottleneck as well as a point of failure if it fills up. For RAID disks we track each partner or member, depending on the architecture.

Continuous improvement comes from recommendations for:

- Rebalancing disk workload by moving files from over-busy disks;
- Adding members to striped disk sets;
- Reducing the number of indexes on files;
- Partitioning files.

We also can track critical file usage, and project and predict if a file is inadequately allocated to handle the anticipated peak day.

Network:

Continuous improvement comes from capturing, filtering, and charting the data for every LAN and serial communications device on the system. Communications is always a critical challenge for performance management. A bottleneck on a comm line is usually a high-impact problem.

In practice, the number and diversity of the communications lines to and from a system means that it is very difficult for a planner to capture traffic data.

The Ban Bottlenecks methodology, when the computer's architecture allows it, will capture throughput data from every comm line on the system. We then filter and chart that data, and relate it to the workload.

Continuous improvement comes from our ability to:

- Detect unusual activity on the comm system, possibly conflicting with transaction traffic;
- Detect when a comm line is "topping out," approaching its peak or practical throughput limit;
- Detect when one or more of a bank of lines are not load-sharing properly;
- Project what the impact of the upcoming peak demand will be on the throughput of the line.

Enterprise:

Continuous improvement comes from comparing and contrasting groups or clusters of computers with or against each other. Ban Bottlenecks provides "group diagnostic reports" for the Enterprise which highlight which systems of a set are having problems. This allows the system manager to zero in on the items which need attention. It also allows the system manager to confirm that load-balancing is operating correctly.

Executive Summary:

Since we have sophisticated techniques to capture, filter, and present the data, we also have the ability to produce executive-level summary information with custom reports, charts, and tables designed to the customer's specifications.

Bottom-Line Value:

The bottom-line value of the Ban Bottlenecks service is not the data collection. It's not the daily and weekly on-system reports. It's not even the analyst-generated Ban Bottlenecks report for your enterprise which covers everything. It is the fact that we review everything that we generate for you, make judgments about it based on our knowledge of your environment, grade it, and then explain it in a monthly web conference so that you don't have to scan all the detail which we provide.

Summary:

Continuous improvement of the computer environment comes from the Ban Bottlenecks methodology. Ban Bottlenecks is the best practice for performance and capacity auditing of a computer system. It is the most comprehensive review of a system and includes most factors that are normally ignored in actual practice. The end result of the Ban Bottlenecks service is a well-performing system adequately prepared to handle anticipated peaks in demand.

Ban Bottlenecks™ is available on the following architectures:

- Stratus VOS
- HP NonStop™
- Unix
- Windows

For further information please contact us at:

1-415-256-8369
inform@banbottlenecks.com
www.banbottlenecks.com