Measuring Resiliency of IT Systems

Aaron B. Brown[1] and Peter Shum[2]
[1]IBM T.J. Watson Research Center (Hawthorne, NY),
abbrown@us.ibm.com
[2]IBM Toronto Laboratory (Markham, ON, Canada)
shum@ca.ibm.com

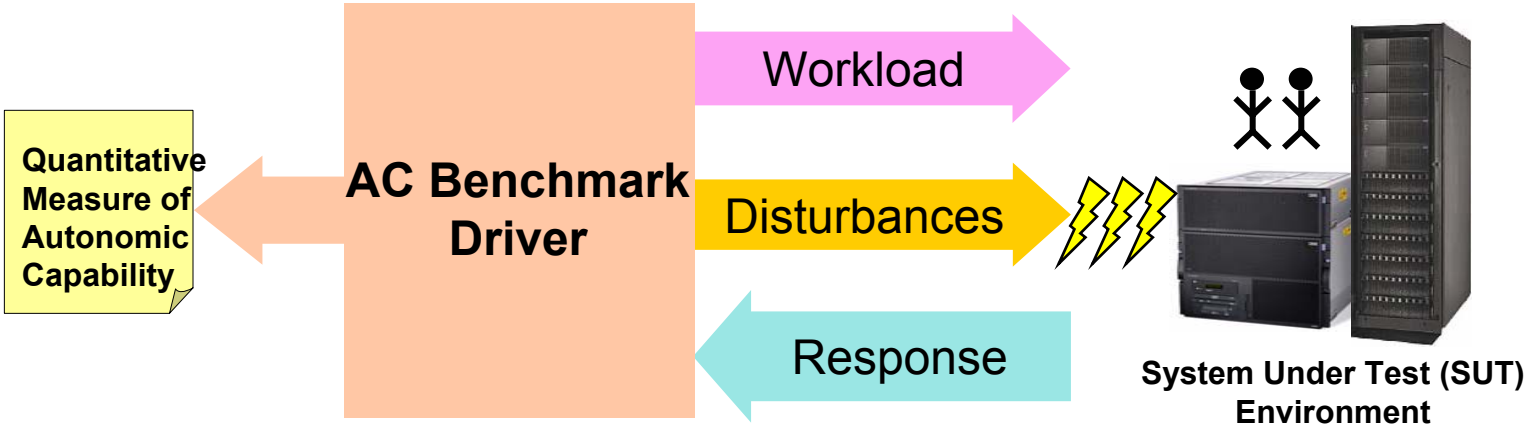
- **Overview**
  - Implementation of and experience with a production-grade benchmark to measure the system resiliency capability of enterprise environments
  - Part of the larger IBM Autonomic Computing Benchmark effort

Problem Statement

"Benchmarks shape a field, for better or worse."
— *Dave Patterson*

- **Benchmarks are essential for quantifying and guiding progress in Autonomic Computing (AC)**
  - Prior work has proposed "AC Benchmarks" focused on the 4 dimensions of AC capability
    - Self-Healing, Self-Configuring, Self-Optimizing, Self-Protecting
- **We have implemented an industrial-grade resiliency benchmark that measures AC Self-Healing / Resiliency capability**
  - Integrates measurement of fault-tolerance/dependability with measurement of autonomic maturity

Benchmarking Autonomic Computing: The Basics



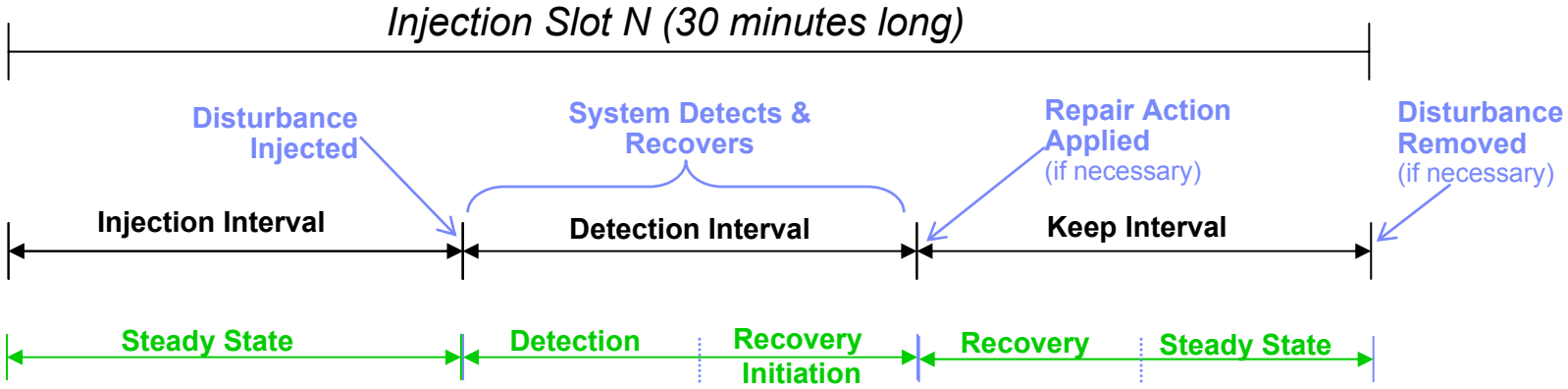
- **Our benchmark quantifies resiliency by measuring a system's response to disturbances in its environment**
  - "Disturbance" could be faults, event, … anything that could change the state
  - Impact of disturbance on Quality of Service (QoS)
  - Ability to adapt effectively to disturbance
  - Degree of automation in response to disturbance
- **Measure resiliency, not availability**
- **Benchmarking approach similar to DBench-OLTP**

Comparison with Dbench-OLTP

	AC Benchmark	DBench-OLTP
Test Target	Multi-components solution e.g. J2EE environment	Single component emphasis e.g. DBMS
Quantitative Metric	Throughput Index	Tf, $/Tf, Ne, AvtS, AvtC
Qualitative Metric	Maturity Index	N/A
Faults/Disturbances	▪ Operator ▪ Attack/Load Surge ▪ Systems (resource, restart loop,..)	▪ Operator ▪ Software ▪ Hardware
Sample Workload	SPECjAppServer2004, Trade6, plug-in workloads with RPT and WSWS	TPC-C
Implementation	Install Shield, Eclipse-based, HTML report, customizable for faults, workloads, and workload drivers	Scripts
Methodology	Similar to DBench-OLTP	DBench-OLTP

Key Aspect: Injecting Disturbances

- **Each disturbance is injected in an Injection Slot while the workload is applied**



- **Injection slots are run back-to-back, preceded by an optional Startup Interval for ramp-up**
- **For disturbances that require human intervention to recover:**
  - The detection interval is replaced by a fixed, 10-minute time penalty
  - Shorter interval for system that auto-detects but requires manual recovery
  - A scripted Repair Action is applied after the detection interval

Disturbances Injected

- **Benchmark capable of injecting 30 types of disturbances**
  - Representing common expected failure modes, based on internal expertise, data, and customer survey
- **Disturbance types**
  - **Attacks** (e.g. runaway query, load surge, poison message)
  - **Unintentional operator actions** (e.g. loss of table/disk, corrupted data file)
  - **Insufficient resources / contention** (e.g. CPU, memory, I/O, disk hogs)
  - **Unexpected shutdowns** (e.g. OS shutdown, process shutdown)
  - **Install corruptions** (e.g. Restart failures on OS, DBMS, App Server)
- **Targeted at OS, all middleware and server tiers, and application**
- ***We are developing more disturbances to expand the benchmark's capabilities for self-healing and beyond***

Top Customer Pains Overall

Customer Pain
Hang failure of a server: database (DBMS)
Application-related hangs: internal application hang
Leaks: memory leak in user application
Database-related data loss: storage failure affecting database data
Restart failure of operating system on: database (DBMS) node
CPU resource exhaustion on: database (DBMS) node
Miscellaneous hang failures: hang caused by unavailability of remote resource (e.g., name/authentication/directory server)
Miscellaneous Restart Failures: orphaned process prevents restart
Restart failure of server process for: database (DBMS) node
Restart failure of operating system on: application server node
Surges: load spike that saturates application
Miscellaneous stops: Unexpected stop of user application
Database-related data loss: loss of an entire database file
Application performance affected due to: parameter setting on database

Metrics for Quantifying Effects of Disturbances (1)

- **Metric #1: Throughput Index**
  - Quantitative measure of Quality of Service under disturbance
  - Similar to typical dependability benchmark measure
  - Computation for disturbance i:

$$ThroughputIndex_i = P_i / P_{base}$$

where

P_i = # of txns completed without error during disturbance injection interval i

P_{base} = # of txns completed without error during baseline interval (no disturbance)

  - Range: 0.0 to 1.0
    - Anything below 0.9 is pretty bad
  - Average over all disturbances to get final score

Metrics for Quantifying Effects of Disturbances (2)

- **Metric #2: Maturity Index**
  - Novel, qualitative measure of degree of Autonomic capability
  - Each disturbance rated on 0 – 8 point scale aligned with IBM's Autonomic Maturity model

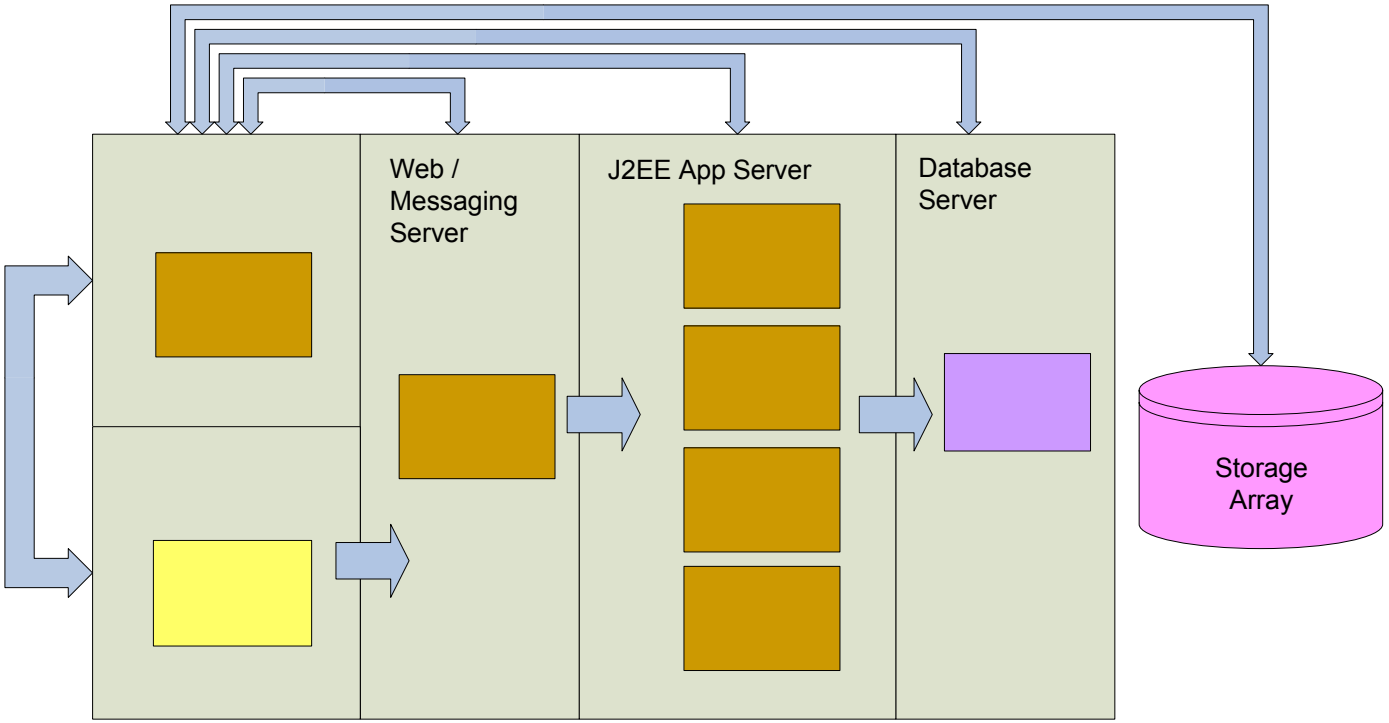
Maturity Level	Brief Description	Points
Basic	*IT staff relies on reports, docs, and manuals to manage individual IT components*	0
Managed	*IT staff uses management tools providing consolidated IT component management*	1
Predictive	*Components monitor and analyze themselves and recommend actions to IT staff*	2
Adaptive	*IT components monitor, analyze, and take action independently and collectively*	4
Autonomic	*IT components collectively & automatically self-manage according to business policy*	8

    - Non-linear point scale gives extra weight higher maturity

  - Ratings based on 90-question survey completed by benchmarker
    - Evaluate how well the system detects, analyzes, and recovers from the failure
    - Example: for abrupt DBMS shutdown disturbance:
      "How is the shutdown detected?
      A. The help desk calls operators to tell them about a rash of complaints (0 points)
      B. The operators notice while observing a single status monitor (1 point)
      C. The autonomic manager notifies the operator of a possible problem (2 points)
      D. The autonomic manager initiates problem analysis (4 points)"
  - Overall score: averaged point score / 8
    - Range: 0.0 to 1.0

Example Benchmark Result: Systems Under Test

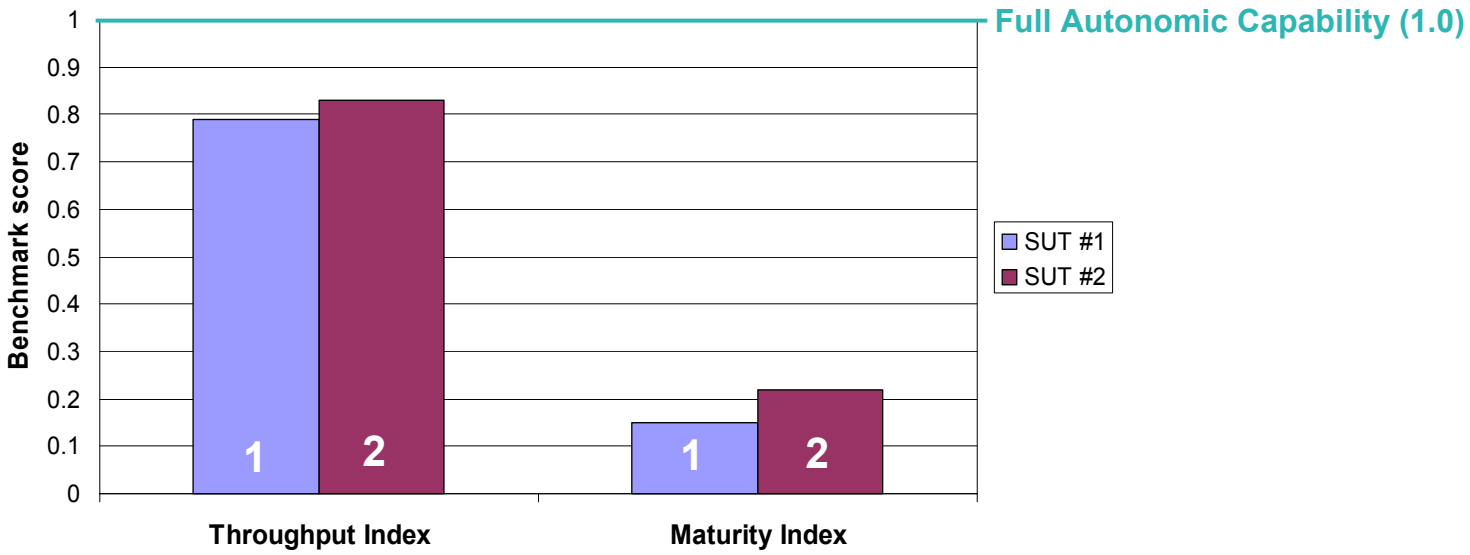
- **Example benchmark run: comparison of two SUTs**
- **SUT #1: complex environment, but no autonomic features:**



- **SUT #2: Identical, but adds a set of system management technologies to be evaluated**

Example Benchmark Result: Summary Scores

- **Summary benchmark results from SUT 1 and 2:**

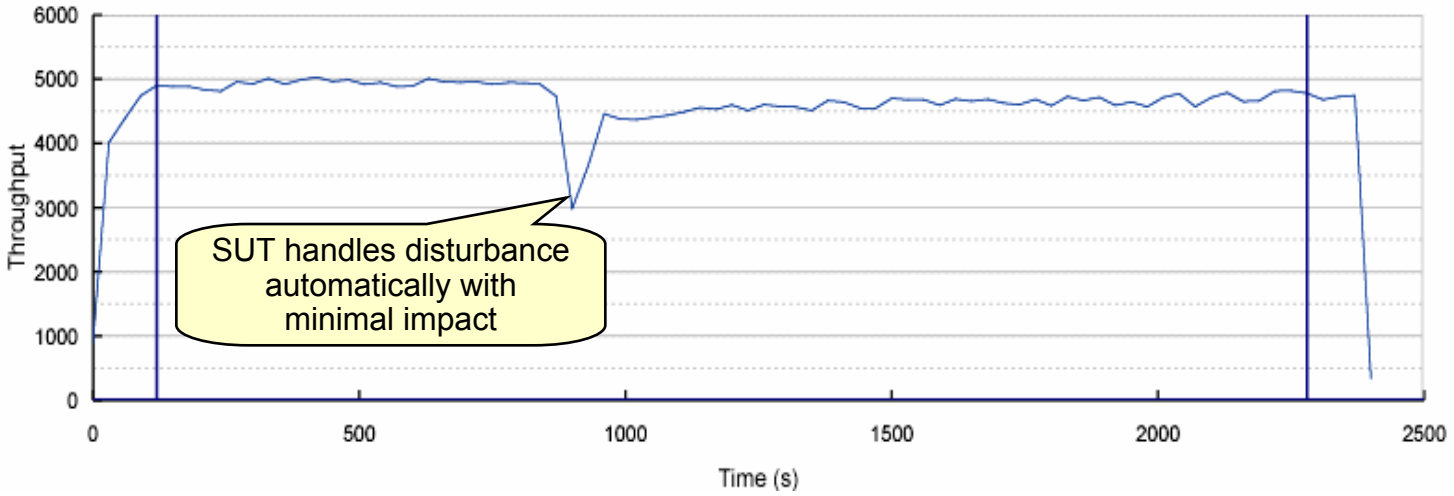


  - Neither SUT scores well on throughput measure or autonomic maturity measure
  - System management technology in SUT #2 provides small but measurable improvement
  - Benchmark results indicate that significant improvements are needed for these SUTs to reach full autonomic maturity for resiliency

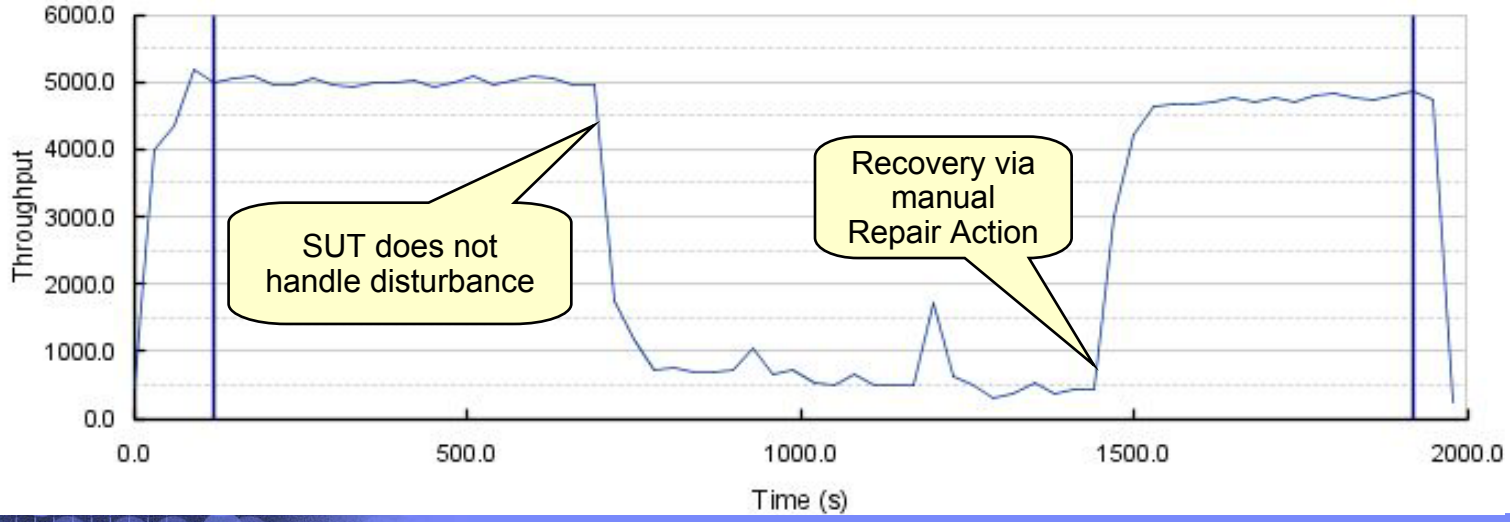
Example Results: Detailed Disturbance Response

- **Comparison of throughput over injection slot for 2 disturbances:**

Disturbance #1
(OK response)



Disturbance #2
(poor response)

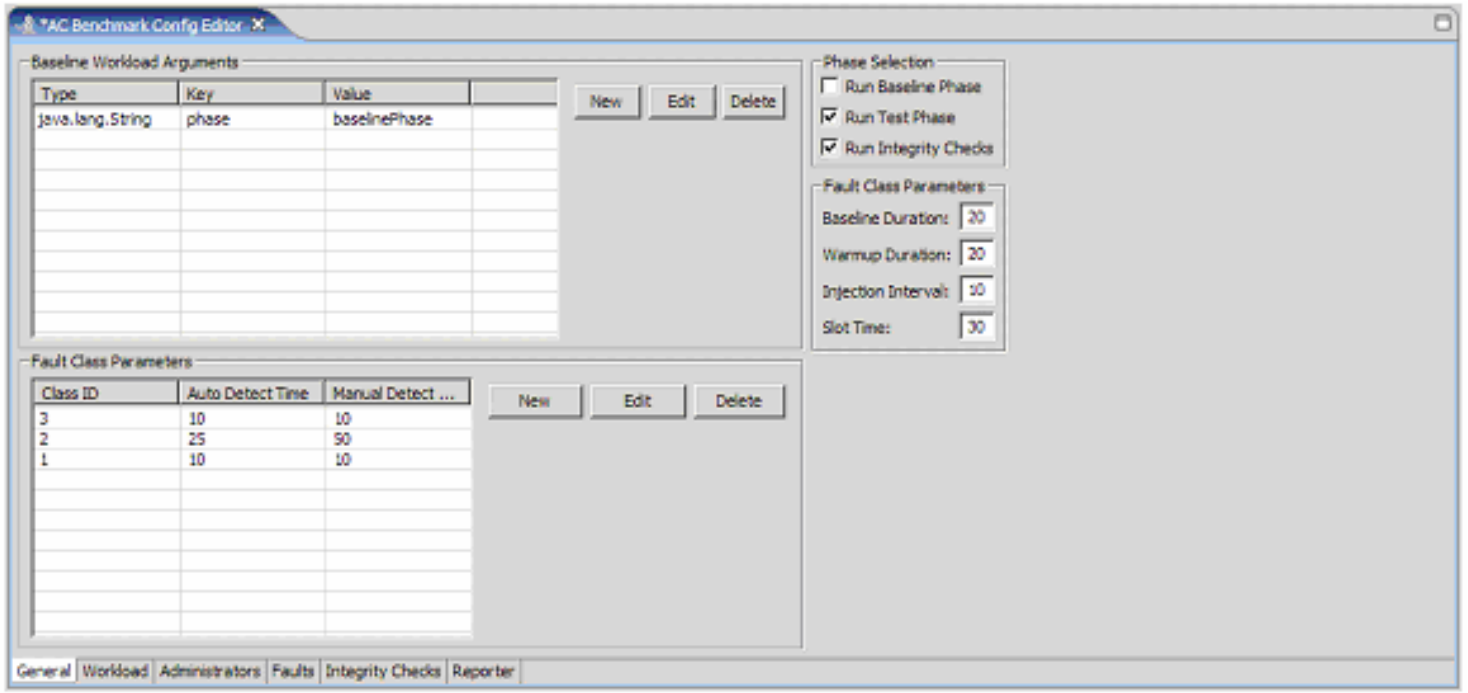


Implementation of the AC Benchmark Kit

- **Implement the resiliency methodology into a benchmark kit**
- **Benchmark kit is targeted at an enterprise multi-tier environment and can be extended to other workloads**
- **Implemented as Eclipse plug-in with GUI and command line**
- **Team has demonstrated portability to**
  - Various workloads
    - SPECjAppServer2004 – a popular standard J2EE benchmark
    - Trade6 - a popular WebSphere J2EE workload
    - TPC-C - (in-progress) a popular OLTP workload for the DBMS
  - Various workload drivers
    - Rational Performance Tester (RPT)
    - WebSphere Studio Workload Simulator (WSWS)

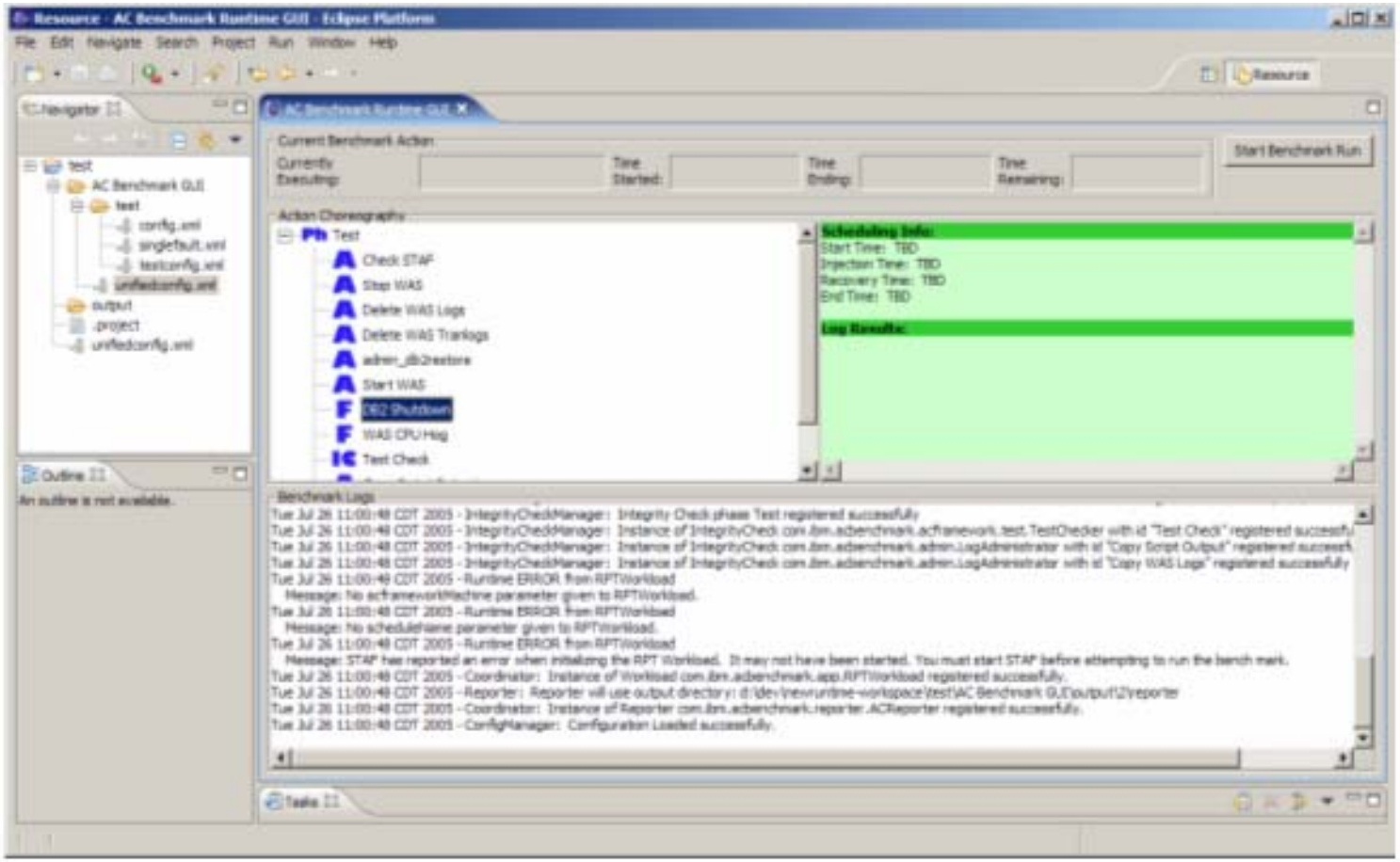
Configuring the AC benchmark

- All configuration information contained in a single xml file
- Can be edited by hand or using an eclipse configuration GUI



Running the benchmark and interpreting the results

- Can be run either from an eclipse runtime GUI or from the command line



Sample html report for a single fault

Autonomic Computing Benchmark

Baseline phase summary

Measurement interval			Response time			Total operations			Results
Start (s)	End (s)	Duration (s)	Min (s)	Max (s)	Mean (s)	Attempted	Successful	Success rate	Throughput (ops/s)
300.0	1500.0	1200.0	N/A	N/A	0.029	N/A	1598387.4	N/A	1331.99

Fault summary

Fault name

Throughput

Fault info		Measurement interval			Response time			Total operations			Results	
Number	Name	Start (s)	End (s)	Duration (s)	Min (s)	Max (s)	Mean (s)	Attempted	Successful	Success rate	Throughput (ops/s)	Throughput index
1	IO hog DB2	902.39	3300.01	2397.62	N/A	N/A	0.039	N/A	2797602.33	N/A	1162.82	0.873

Graphs

- Baseline phase: Pages per second
- Baseline phase: Page elements per second
- Baseline phase: Average response time
- Baseline phase: MB read
- Baseline phase: MB written

Graphs for the baseline phase

- IO hog DB2: Pages per second
- IO hog DB2: Page elements per second
- IO hog DB2: Average response time
- IO hog DB2: MB read
- IO hog DB2: MB written

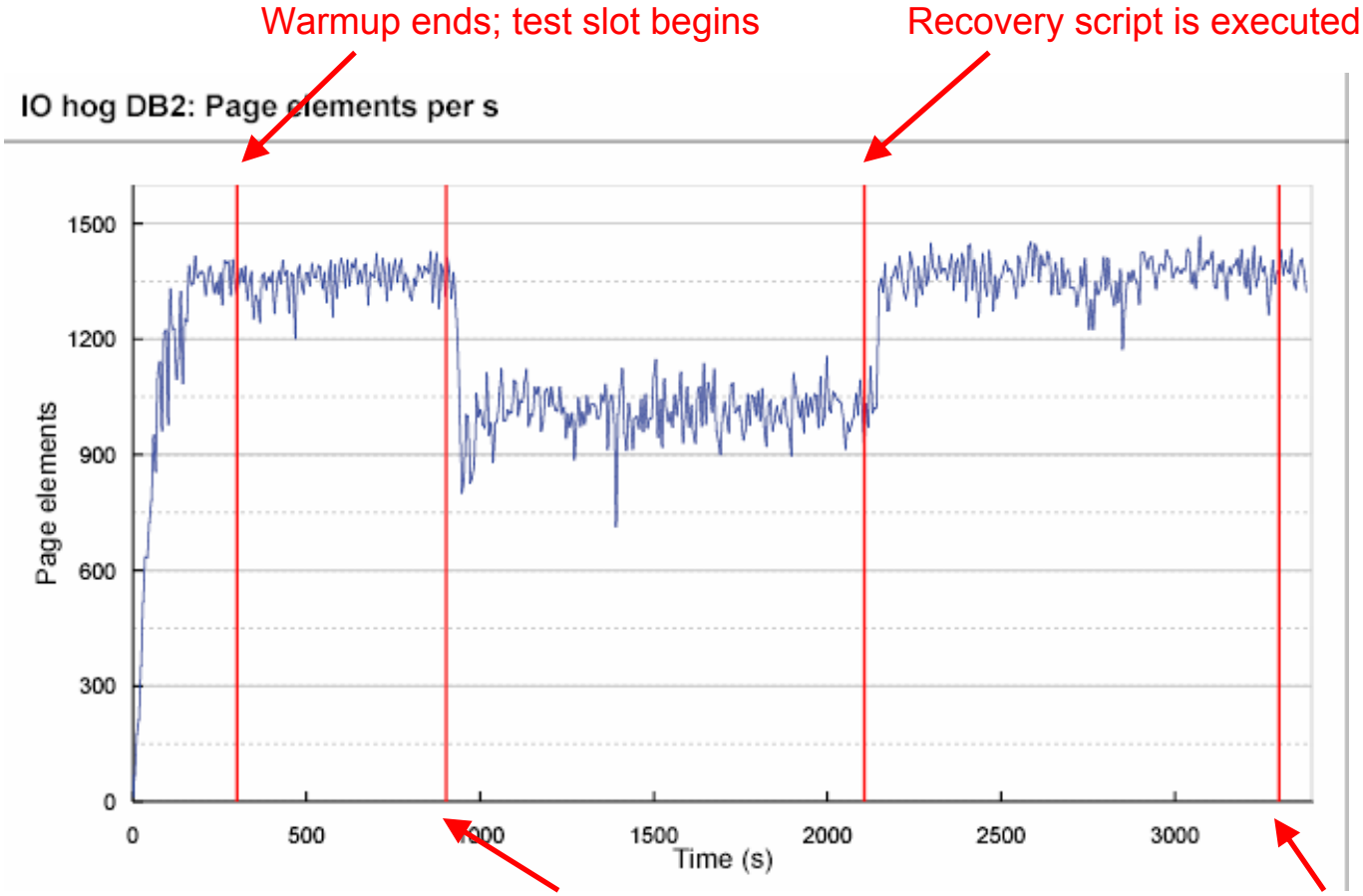
Graphs for the fault

Throughput index
= throughput (test) /
throughput (baseline)

Run conditions -- baseline phase

- Script : /root/automation_ver2/Benchmarks/trade6/jibe/trade6_rand.jxs
- Initial number of clients : 30.0
- Run time : 00:25:21

Sample throughput graph for a single fault



Sample html report for four faults run consecutively

Autonomic Computing Benchmark

Baseline phase summary

Measurement interval			Response time			Total operations			Results
Start (s)	End (s)	Duration (s)	Min (s)	Max (s)	Mean (s)	Attempted	Successful	Success rate	Throughput (ops/s)
600.0	1800.0	1200.0	N/A	N/A	0.03	N/A	1203371.0	N/A	1002.81

Fault summary

Fault info		Measurement interval			Response time			Total operations			Results	
Number	Name	Start (s)	End (s)	Duration (s)	Min (s)	Max (s)	Mean (s)	Attempted	Successful	Success rate	Throughput (ops/s)	Throughput index
1	CPUhog on WAS appserver	1200.66	2400.02	1199.36	N/A	N/A	0.033	N/A	1075194.67	N/A	896.47	0.894
2	Deadlock DB2	3002.88	4200.66	1197.78	N/A	N/A	0.029	N/A	1203550.0	N/A	1004.82	1.0
3	Disk hog on DB2	4804.53	6002.27	1197.73	N/A	N/A	0.028	N/A	691198.0	N/A	577.09	0.575
4	Network down WAS	6606.95	7804.72	1197.77	N/A	N/A	0.456	N/A	369251.0	N/A	308.28	0.307

Graphs

- Baseline phase: Pages per second
- Baseline phase: Page elements per second
- Baseline phase: Average response time
- Baseline phase: MB read
- Baseline phase: MB written

Graphs for the baseline phase

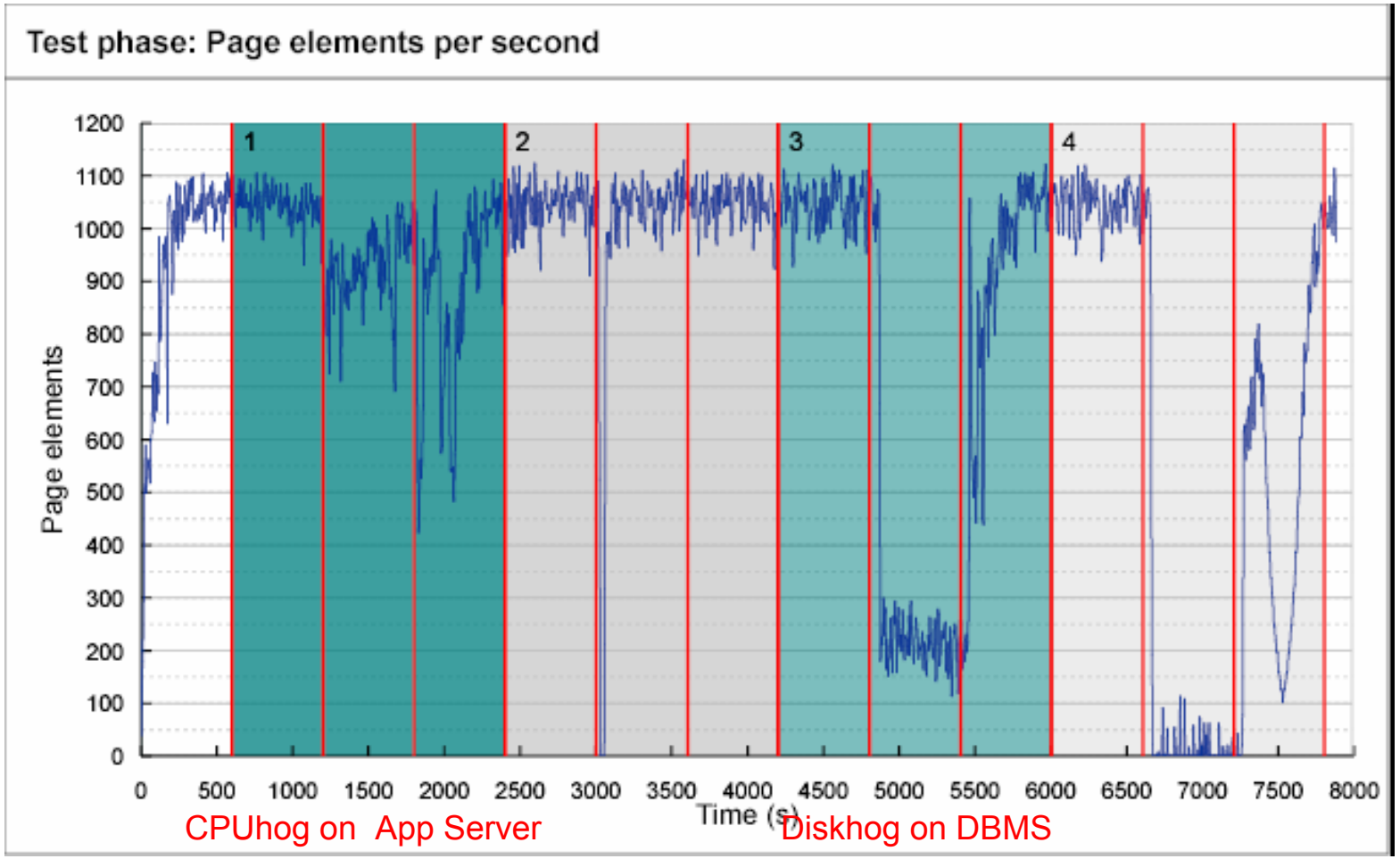
- Test phase: Pages per second
- Test phase: Page elements per second
- Test phase: Average response time
- Test phase: MB read
- Test phase: MB written

Graphs for all four faults together

- CPUhog on WAS appserver: Pages per second
- CPUhog on WAS appserver: Page elements per second

Graphs for each separate fault

Sample graph showing throughput across all four faults



Issues and Challenges for Resiliency Benchmark

- **Accounting for incomplete healing**
  - Need to differentiate bypass vs repair/reintegration
- **Accounting for resources utilization**
  - Need to differentiate efficiency vs over-provisioning
- **Fair comparsion between systems of different scales**
  - High throughput system takes longer to come down, and longer ramp-up to steady state after restart
- **Defining reporting rules**
  - Use simple metric to focus and compare, and report other useful data for analysis

Conclusions

- **We have built the first implementation of a benchmark for System Resiliency capability**
  - Combines a dependability measure (tolerance to disturbances) with a measure of Autonomic Maturity
  - Provides a quantitative way to assess automated resiliency of IT systems
  - Targeted at enterprise environments, capable of working at enterprise scale
- **Sample results & internal experience illustrate utility of benchmark, and the flexibility and robustness of the benchmark kit**
  - Work with multi-tier components
  - Easily customizable for new faults, workloads, and workload drivers.
- **Though many challenges remain to increase sophistication of this benchmark, the kit provides a robust foundation for future extensions**