Using Fault Injection and Modeling to Evaluate the Performability of Cluster-based Services

Kiran Nagaraja, Xiaoyan Li, Ricardo Bianchini, Richard P. Martin, Thu D. Nguyen

Presented By

Neeraj Krishnan

CS 553, Spring 2003

Authors

Kiran Nagaraja and **Xiaoyan Li**, outstanding graduate students in Computer Science at Rutgers. Research in Distributed Systems, Internet Services, etc.

Dr. Bianchini - Parallel, distributed and cluster computing, power and energy optimization.

Dr. Martin - Scalable Internet Services, Spatial Computing

Dr. Nguyen - Parallel, distributed systems, security

"PANIC"

Main Idea

Propose a **two-phase methodology** for measuring average throughput and availability (in the presence of faults)

Measure performability (performance + availability)

(i.e. come up with a metric)

Use it to compare different versions of Press (cluste based internet service)

Study impact of design decisions

General design guidelines for building highly available systems

Motivation

Understanding design for availability

Cluster **Behaviour under faults**

Performance Vs Availability (Inherent contradiction ?)

What about non-cluster systems?

Applicable to all cluster systems?

What we'll do

2-phase methodology
- fault injection
- modeling

Case Study - Apply 2-phase methodology to Press

Impact of design decisions and scaling

Conclusion and Contribution

Critique - reviewers comments ! (USITS '03)

5

2-phase Methodology

Phase 1:

Fault Injection

Inject faults and measure performance (throughput)

Phase 2:

Modeling

Use Phase 1 results + fault load + environment parameters to get average throughput and availability.

Get PERFORMABILITY

6

Phase 1: Fault Injection - Mendosus

Mendosus:

A fault injection and network emulation infrastructure

To test a service by injecting faults

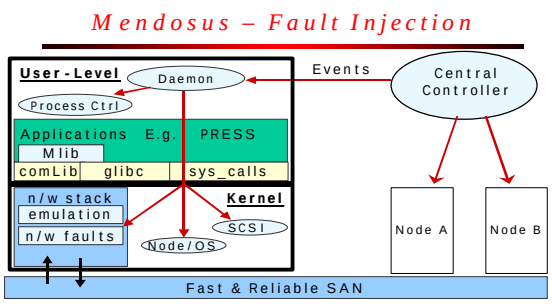
Runs on a cluster of PCs connected to a Giganet VIA network

S/W Components
- Controller
- Daemon
- LAN emulator module
- Fault Injection module

Mendosus: A SAN-Based Fault-Injection Test-Bed for the Construction of Highly Available Network Services
Xiaoyan Li, Richard Martin, Kiran Nagaraja, Thu D Nguyen, Bin Zhang

7

Phase 1: Fault Injection - Mendosus



8

Phase 1: Fault Injection - Fault Models

Mendosus can insert several kinds of faults, a subset of which is used for this study, and they are

Network: NIC, link, switch down

causes packet loss

Disk: hang, SCSI timeout

causes delays, execution stall

Node: reboot, freeze

Application: hang, crash

Each of these affect throughput in certain ways

Some faults are masked to a certain extent

2-phase Methodology

Phase 1:

Fault Injection

Inject faults and measure performance (throughput)

Phase 2:

Modeling

Use Phase 1 results + fault load + environment parameters to get average throughput and availability.

Get PERFORMABILITY

Phase 2: Modeling - Assumptions

Assumptions:

No correlated faults

Fault arrivals are exponentially distributed

No overlapping faults -> Faults queue at the system

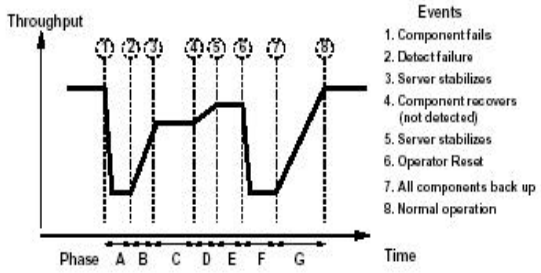
Phase 2: Modeling - Fault Model

The seven stage fault model

piecewise linear

for each stage we have duration and average throughput

Phase 2: Modeling - Fault Model



Phase 2: Modeling - Fault Model

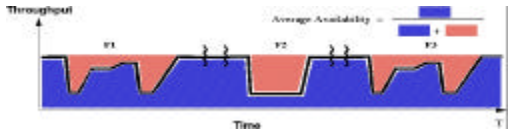
$$AT = (1 - \sum_c W_c)T_n + \sum_c \sum_{s=A}^{G} (\frac{D_c^s}{MTTF_c} T_c^s)$$

$$AA = \frac{AT}{T_n}$$

where $W_c = (\sum_{s=A}^{G} D_c^s)/MTTF_c$.

Phase 2 – Modeling – Fault Model

Computing Average Availability



- Assumptions:
  - Faults are non-overlapping and independent
- Parameters: MTTF, MTTR
  - Sources: [Sullivan91, Chillarege95, Iyer99, Talagala99, Trivedi00, Heath02]
- Measure throughput under single fault

Phase 2: Modeling - PERFORMABILITY

$$P = T_n \times \frac{\log(A_I)}{\log(AA)}$$

Phase 2: Modeling - PERFORMABILITY

Essentially:

$P \propto T_n$

$P \propto 1/u$ (u is unavailability)

$P = k (T/u)$

But in the manner given, the choice of the proportionality constant is made easy to pick

Phase 2: Modeling - PERFORMABILITY

What's the significance of the log scale ?

Does the absolute value of performability mean anything ?

The number of orders of magnitude that you are off from peak throughput on account of unavailability

What we'll do

2-phase methodology

- fault injection
- modeling

Case Study - Apply 2-phase methodology to Press

Impact of design decisions and scaling

Conclusion and Contribution

Critique - reviewers comments ! (USITS '03)

Case Study

Press:

Cluster-based **Locality conscious** web server

Different versions

- cooperative/non-cooperative
- user/kernel level communication

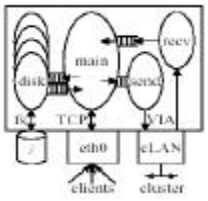
Phase 1:

Behaviour under faults

Phase 2:

Use modeling to get performability

Case Study - Press



Case Study - Press

Version	Main Features	Expected Behavior	Throughput
I-PRESS	Independent servers	Lowest performance but most robust to failures.	1298 req/sec
TCP-PRESS	Cooperative caching servers using TCP for intra-cluster communication	Performs better than independent versions due to higher cache hit rates. Introduces dependencies between nodes, so a failure may affect multiple nodes. If fault leads to permanent break in communication or loss of one or more PRESS processes, then must be restarted manually by operator for full recovery.	4965 req/sec
ReTCP-PRESS	Cooperative caching servers using TCP for intra-cluster communication and dynamic reconfiguration	Performs as well as TCP-PRESS but should be able to recover automatically from most faults.	4965 req/sec
VIA-PRESS	Cooperative caching servers using VIA for intra-cluster communication and dynamic reconfiguration	Performs the best and, like ReTCP-PRESS, should be able to automatically recover from more faults than TCP-PRESS.	6878 req/sec

Case Study

Press:

Cluster-based **Locality conscious** web server

Different versions

- cooperative/non-cooperative
- user/kernel level communication

Phase 1:

Behaviour under faults

Phase 2:

Use modeling to get performability

Case Study - Phase 1

RECAP:

Network: NIC, link, switch down

causes probabilistic packet loss

Disk: hang, offline, power failure, read, write fault, SCSI timeout, parity errors, bus busy, queue full

causes delays, execution stall

Node: hard and soft reboot, freeze

Application: hang, crash

Each of these affect throughput in certain ways

Case Study - Phase 1

This is how the different Press versions behave under these faults:

The independent version is entirely predictable (no cooperation effects), throughput is always proportional to number of servers up

What are the interesting cases?

25

Case Study - Phase 1 - Network Faults

switch, nic, link failure

TCP: Switch fails -> queues back up -> 0 thruput until switch recovers

ReTCP: Switch fails -> queues back up -> 0 thruput till detection -> 4 singleton groups -> no reconfiguration !

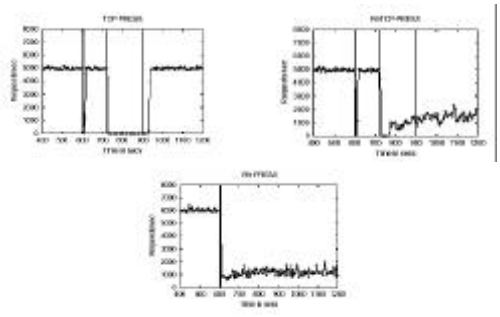
VIA: Switch fails -> queues back up -> immediate detection -> 4 singleton groups -> no reconfiguration !

Nodes haven't crashed so you don't send out new group join message

Fault model mismatch !

26

Case Study - Phase 1 - Switch Failure



27

Case Study - Phase 1 - Disk Faults

SCSI t/o, hang, read/write faults

TCP: t/o -> disk threads blocked -> disk-to-main threads backup -> main thread blocks -> 0 thruput -> recovery -> normal thruput

ReTCP: t/o -> disk threads blocked -> disk-to-main threads backup -> main thread blocks -> 0 thruput -> node with disk fault excluded -> singleton + 3-group -> doesn't regain normal thruput

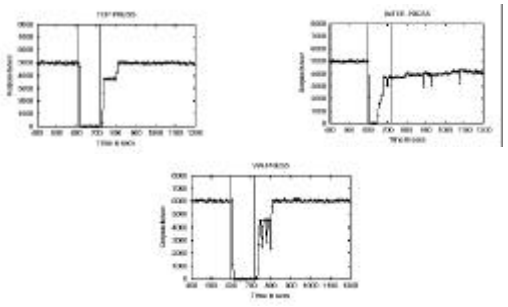
VIA: t/o -> disk threads blocked -> disk-to-main threads backup -> main thread blocks -> 0 thruput -> recovery -> normal thruput

ReTCP interpreted disk hang as node crash

Fault Model mismatch !

28

Case Study - Phase 1 - SCSI timeout



29

Case Study - Phase 1 - Node Faults

crash, freeze

TCP: crash -> queues backup -> 0 thruput -> reboot -> TCP conn breaks -> 3-group + singleton

ReTCP: crash -> queues backup -> 0 thruput -> 3-node cluster -> reboot -> reintegrated -> normal throughput

(this is what ReTCP was designed for)

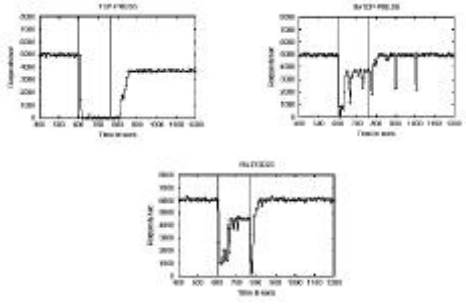
VIA: crash -> queues backup -> immediate detection -> 3-node cluster -> reboot -> reintegrated -> normal throughput

But a freeze is not a crash so ReTCP is once again confused

Fault Model mismatch !

30

Case Study - Phase 1 - Node Crash



31

Case Study - Phase 1 - Application Faults

crash, hang

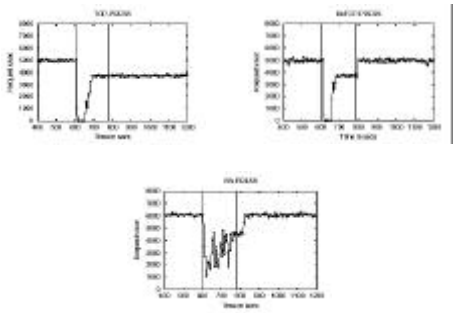
just like node crash, except TCP recovers faster because TCP connection is broken

but a hang is not a crash, so ReTCP is (once again) confused

Fault Model mismatch !

32

Case Study - Phase 1 - Application Crash



33

Case Study

Press:

Cluster-based **Locality conscious** web server

Different versions

- cooperative/non-cooperative
- user/kernel level communication

Phase 1:

Behaviour under faults

Phase 2:

Use modeling to get performability

34

Case Study - Phase 2

We have throughput figures from phase 1

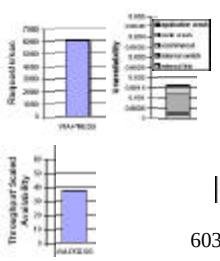
We will use MTTF and MTTR values for the different faulty components

Plug in the values to get AT, AA and P

We shall see it for the VIA version first, then compare all versions

35

Case Study - Phase 2



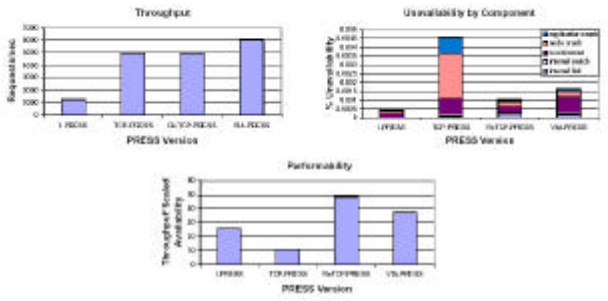
Phase	Switch Failure	
	Throughput (req/sec)	Duration (sec)
A	892.40	75
B	–	0
C	1106.30	3525
D	–	0
E	1259.60	300
F	0.0	300
G	3017.00	300

Switch down	MTTF 1 year	MTTR 1 hour

6031 * log(0.99999)/log(1 - 0.0016)

36

Case Study - Phase 2



What we'll do

2-phase methodology

- fault injection
- modeling

Case Study - Apply 2-phase methodology to Press

Impact of design decisions and scaling

Conclusion and Contribution

Critique - reviewers comments ! (USITS '03)

Studying Impact of Design Decisions

Operator delay (Duration of phase E)

MTTF (e.g. RAIDs have higher MTTFs)

You can also extend it to evaluate impact of using faster processors, more memory, etc

Studying Impact of Scaling

Effect on normal throughput

Effect on MTTF

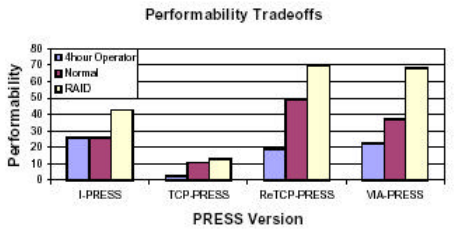
Effect on durations of phases

Effect on fault classes

Infact the scaling study looks at 8 nodes (and memory scaled/not scaled)

You see the impact of disk faults almost vanishing since the whole working set fits in memory

Studying Impact of Design Decisions



41

What we'll do

2-phase methodology
- fault injection
- modeling

Case Study - Apply 2-phase methodology to Press

Impact of design decisions and scaling

Conclusion and Contribution

Critique - reviewers comments ! (USITS '03)

42

Conclusion and Contribution

Methodology to measure average throughput and availability

Metric for performability

Fault Model mismatch and effects

Sets it up for FME and other improvement techniques

43

What we'll do

2-phase methodology
- fault injection
- modeling

Case Study - Apply 2-phase methodology to Press

Impact of design decisions and scaling

Conclusion and Contribution

Critique - reviewers comments ! (USITS '03)

44

Critique and Food for Thought

- Does fault injection uncover the kinds of faults that occur in real systems ?
- Operating under 90% load at all times ?
- Availability as time averaged throughput ?
- Performability ?
- Are conclusions really general or is it peculiar to a "quirky" system like PRESS ?
- Representative of all cluster-based services ?

45

Applause

46