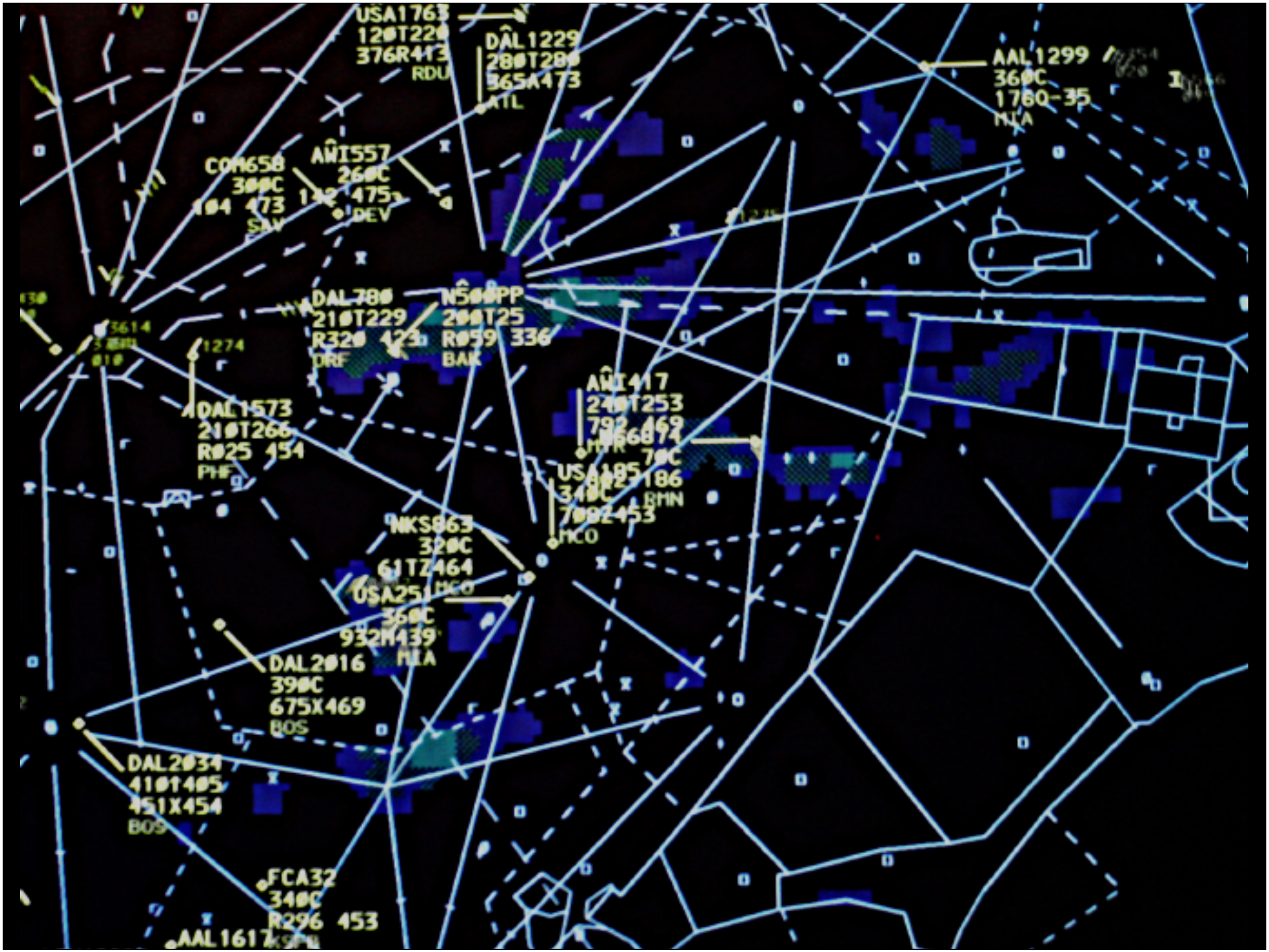
QCon London 2009

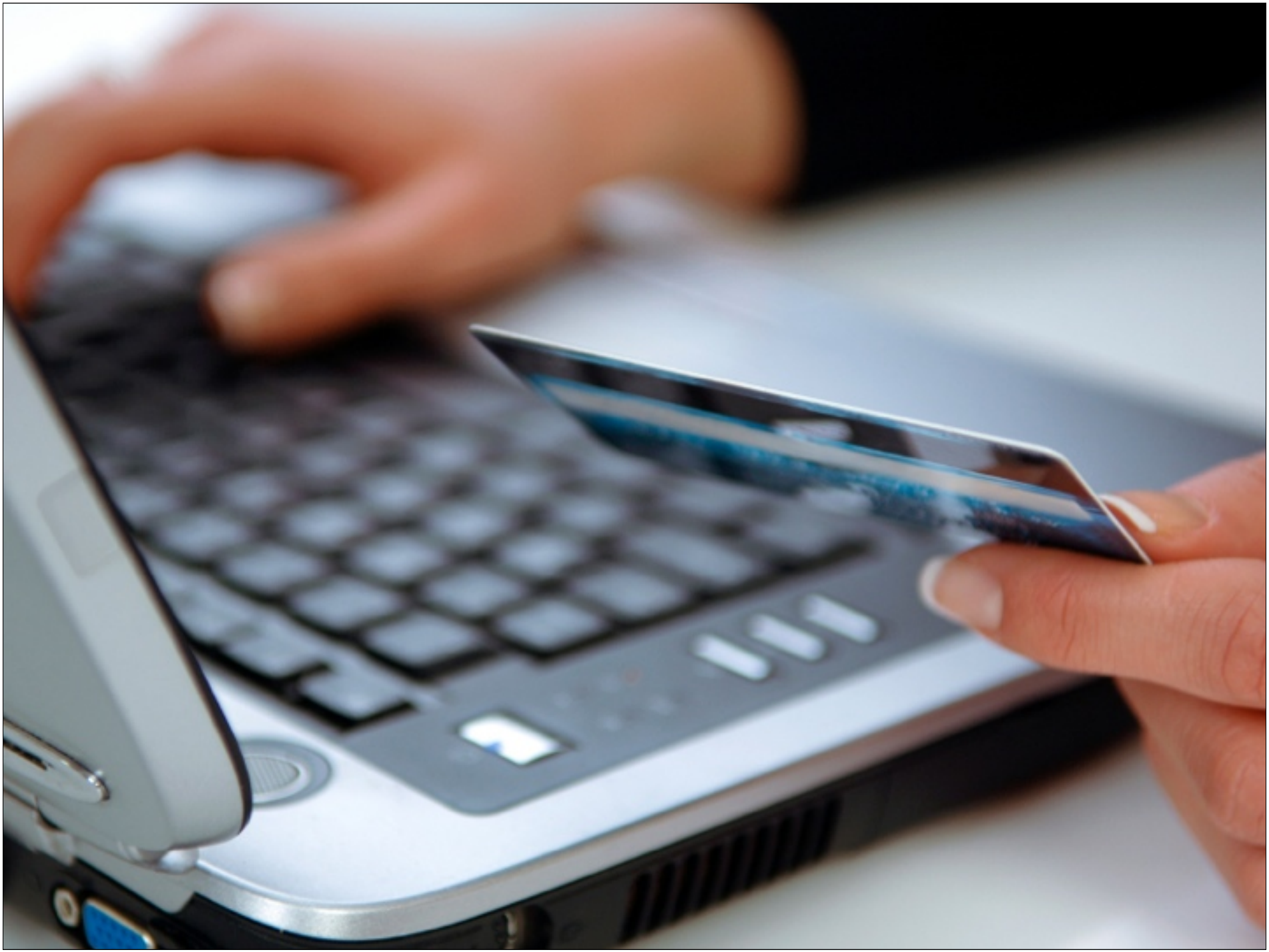
The Realities of Continuous Availability

Mark Richards
Director and Sr. Architect, Collaborative Consulting, LLC
Author of Java Transaction Design Strategies (C4Media)
Author of Java Message Service 2nd Edition (O'Reilly)


USA1763
DAL1229
AAL1299
360C
1760-35
MIA
COM658
300C
SAV
AWI557
260C
DEV
DAL780
210T229
R320 423
DRF
N588PP
200T25
R059 336
BAK
AWI417
240T253
792 469
DAL1573
210T266
R025 454
PHF
NKS863
320C
61TZ464
MCO
USA251
360C
932M439
MIA
DAL2016
390C
675X469
BOS
DAL2034
410T485
451X454
BOS
FCA32
340C
R296 453
AAL1617


5922.56 BUY +04.49
8684.86 BUY +01.58
4283.23 BUY +07.02
4121.06 BUY +03.70
8471.45 BUY +05.26
7518.78 BUY +05.98
1240.92 BUY +01.35
644.04 BUY


1.7810
1.7765
1.7900
1.7855
1.7810
1.7765
08.47
11.12
14.56
19.00
05.14
08.47



WWW.ALGIDACITY.CZ/AUKCE

Internetová Aukční Síň

Unilever

ÚVODNÍ STRÁNKA | GALERIE | VYDRAŽENO | BENEFICE | PRAVIDLA | KONTAKT | switch to ENGLISH

Jste přihlášen(a) jako: **filip** | Váš profil | ODHLÁSIT Z AUKCE

AUKČNÍ SÍŇ

Přispějte i vy na projekty, které společnost Unilever podporuje a získejte hodnotné umělecké dílo. Zaregistrujte se do internetové aukční síně Unilever a získáte tak možnost vydražit si pro sebe či své blízké některý z krásných obrazů od našich předních českých grafiků. eAukce probíhá po dobu 12 měsíců a každý měsíc se draží jiné obrazy.

DETAIL A INFO

Jan Grimm
Tři v kavárně

aktuální částka: **23000 Kč**

historie přihazování:

JAK PŘIHAZOVAT?

− 200 +	PŘIHOZENO 0
− 500 +	ZAPLATÍTE 23000
− 1000 +	
− 5000 +	

☑ Zaškrtněte, chcete-li být na Váš E-mail informován o přebití vaší nabídky

PŘIHODIT

how much availability is “good enough”?

90.0% (one nine)	36 days 12 hours
99.0% (two nines)	87 hours 46 minutes
99.9% (three nines)	8 hours 46 minutes
99.99% (four nines)	52 minutes 33 seconds
99.999% (five nines)	5 minutes 35 seconds
99.9999% (six nines)	31.5 seconds

how much availability is “good enough”?

how about three nines (99.9%)?

there would be a 99.9% turnout of registered voters in an election

if you used your windows pc 40 hours per week, you would only have to reboot it once every two weeks (once a year for a mac )

you would have one rainy day every three years

if you made 10 calls a day you would have 3 dropped calls a year

how much availability is “good enough”?

how about three nines (99.9%)?

the u.s. postal services would lose 2,000 pieces of mail each hour

20,000 prescription errors would be made each year

there would be 500 incorrect surgical operations per week

remember the old days?

availability was handled by large mainframes and fault tolerant systems

hardware and os were extremely reliable and very mature

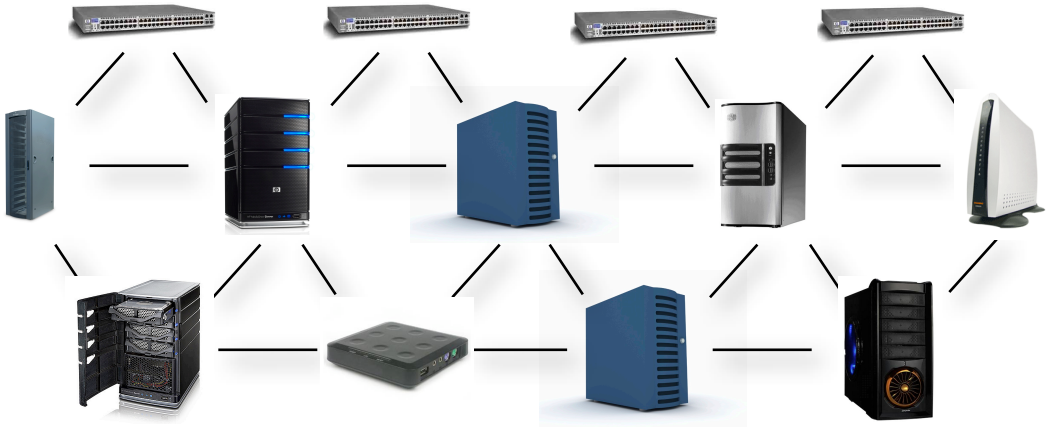
software was thoroughly tested

there were highly trained and skilled operators

redundancy eliminated single points of failure

four nines availability was very common for all aspects of the computing environment

now we have this...



commodity hardware with around 99% availability

short time-to-market requirements usually equates to shortcuts in reliability and system availability design

frequent software changes go largely untested

heterogeneous systems from different vendors making interoperability and monitoring difficult

system complexity and diversity make it difficult to identify the root cause of a failure

system complexity results in faults caused by operator error (over 50% of faults in most cases)

continuous availability

what is it?

high availability

reactive in nature and places an emphasis on failover and recovery in the shortest time possible

continuous availability

proactive in nature and places an emphasis on redundancy, error detection, and error prevention

if this is high availability...

then this is continuous availability

if a tree falls in a forest and no one is around to hear it, does it make a sound?

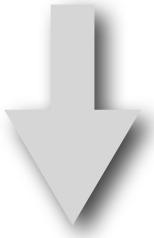
if a fault can be recovered before the user is aware that the fault occurred, is it really a fault?

the fact is, continuous availability systems don't really fail over

"If a problem has no solution, it may not be a problem, but a fact - not to be solved, but to be coped with over time."

- Shimon Peres

continuous availability embraces the philosophy of “let it fail, but fix it fast.”

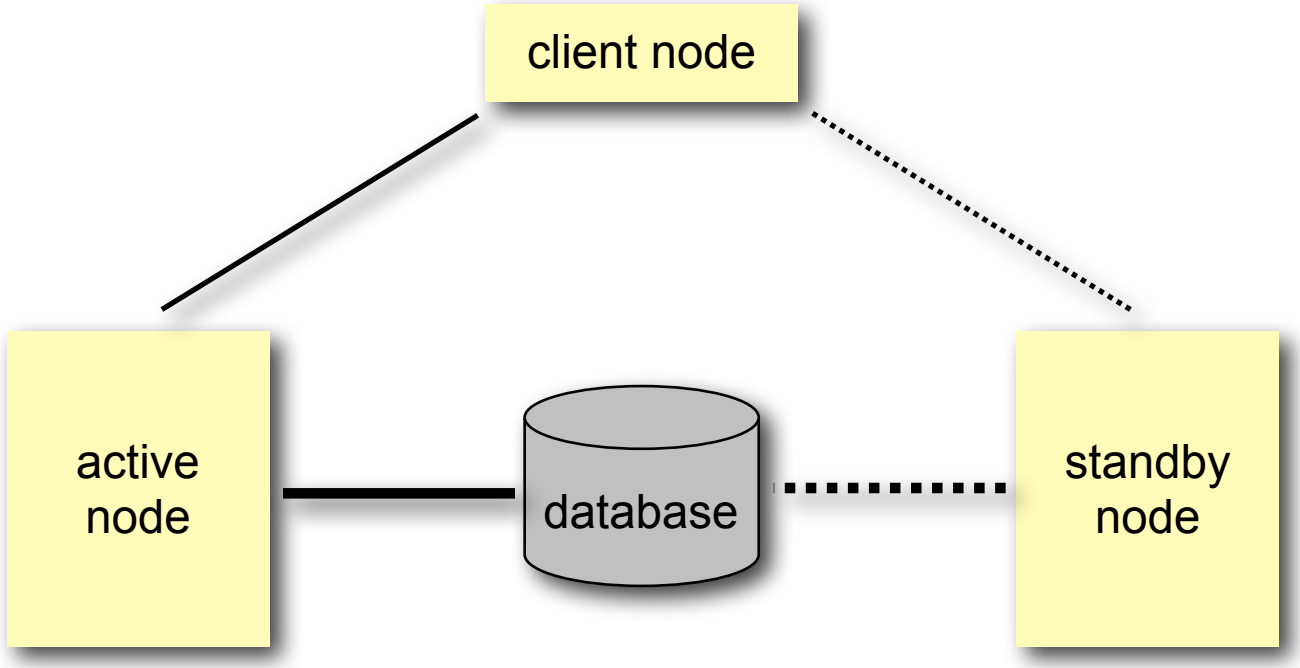


Resubmit rather than fail over

what topologies are needed to support high and continuous availability?

standard high availability topology

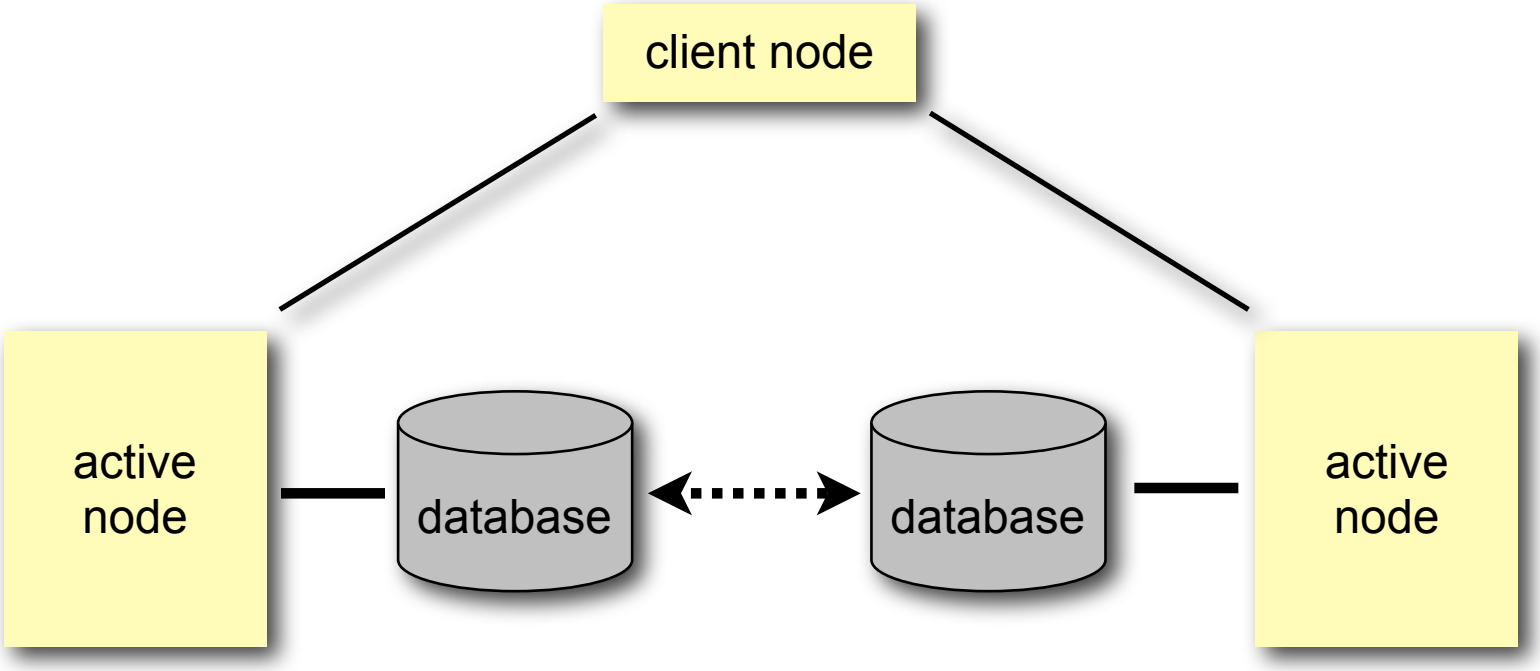
cluster configuration



mean time to failover (mtfo) = **minutes**

standard continuous availability topology

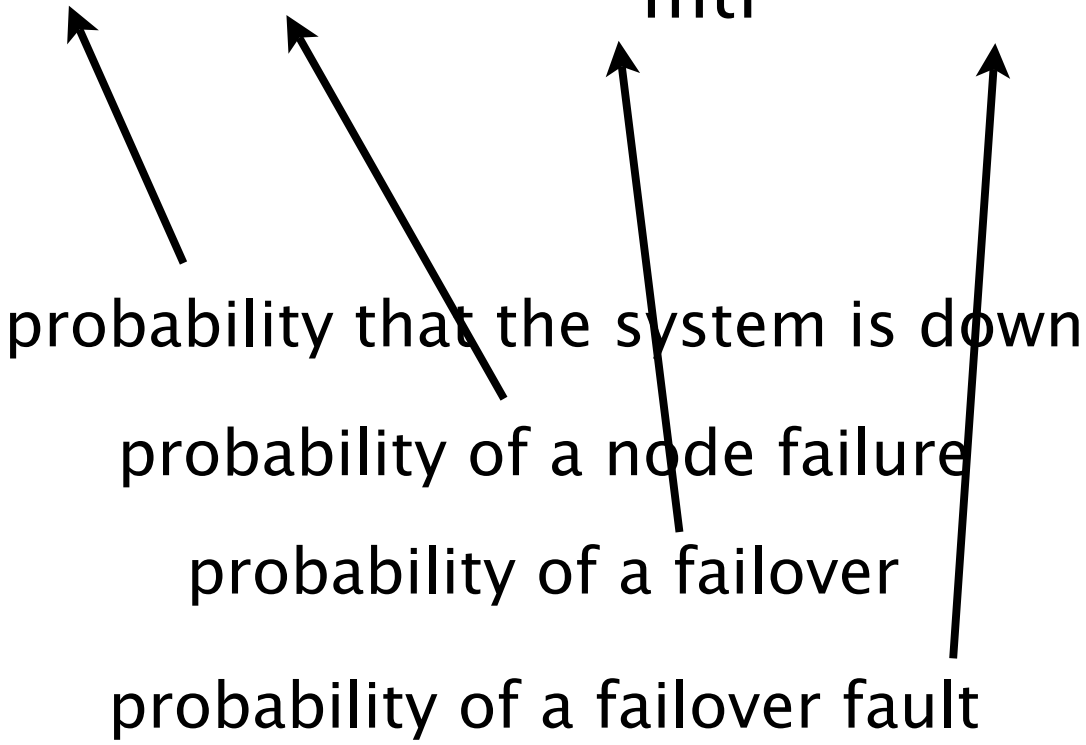
active/active configuration



mean time to failover (mtfo) = **seconds**

calculating system downtime probability

$$sd = (1-a)^2 + (1-a)\frac{mtfo}{mtr} + (1-a)d$$



calculating system downtime probability

$$sd = (1-a)^2 + (1-a)\frac{mtfo}{mtr} + (1-a)d$$

sd = probability of system downtime

a = probability that node is operational

mtfo = mean time to failover

mtr = mean time to repair node

d = probability of a failover fault

let's do the math...

$$sd = (1\text{-}a)^2 + (1\text{-}a)\frac{mtfo}{mtr} + (1\text{-}a)d$$

dual node high availability cluster (active/passive)

a = .999
mtfo = 5 minutes
mtr = 3 hours
d = .01

.000001
+ .00002777778
+ .00001

.000038777778

.9999612222222
or a little under 5 nines
(~ 6 minutes of downtime)

let's do the math...

$$sd = (1\text{-}a)^2 + (1\text{-}a)\frac{mtfo}{mtr} + (1\text{-}a)d$$

dual node continuous availability topology (active/active)

a = .999
mtfo = 3 seconds
mtr = 3 hours
d = 0

.000001
+ .0000002777778
+ 0

.0000012777778

.999998722

or a little under 6 nines
(~ 30 seconds of downtime)

bottom line

clustering = high availability

active/active = continuous availability

the other bottom line

none of this math and theory makes a bit of difference if your application architecture doesn't support the continuous availability environment

continuous availability

a holistic approach

continuous availability killers

id generation or random number generation

processing order requirements

batch jobs and scheduled tasks

application or service state

long running processes and process choreaography

in-memory storage or local disk access

tightly coupled systems

specific ip address or hostname requirements

long running transactions (database concurrency)

but that’s only the start...

most businesses don't really need continuous availability

or do they...

another perspective...

so far the focus has been on system failures

but what about planned outages for maintenance upgrades and application deployments?

issues facing many large companies

increased batch cycles mean longer batch windows

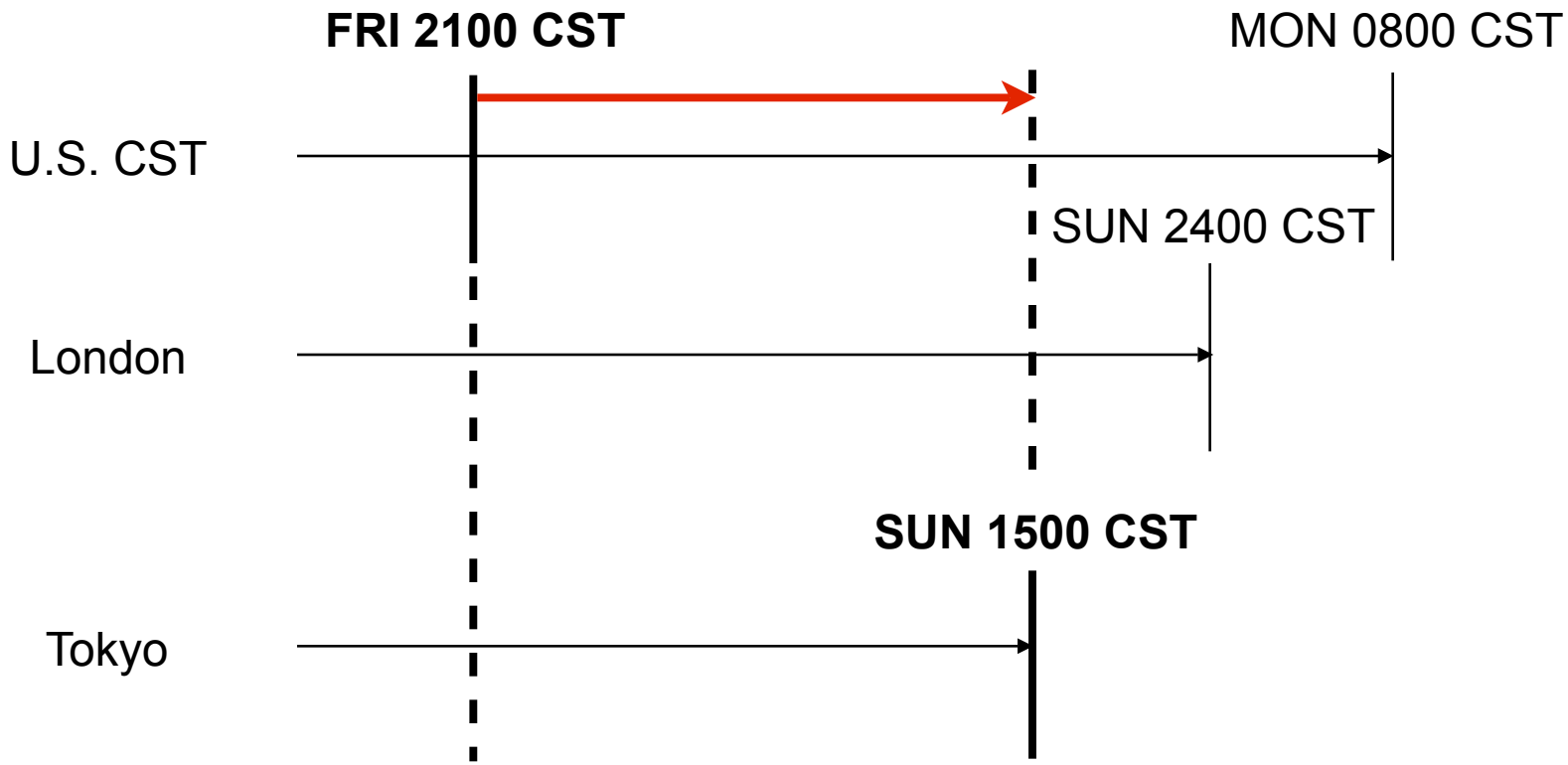
global operations support

increased processing volumes (orders, trades, etc.)

window for applying maintenance upgrades and application deployments is quickly diminishing!

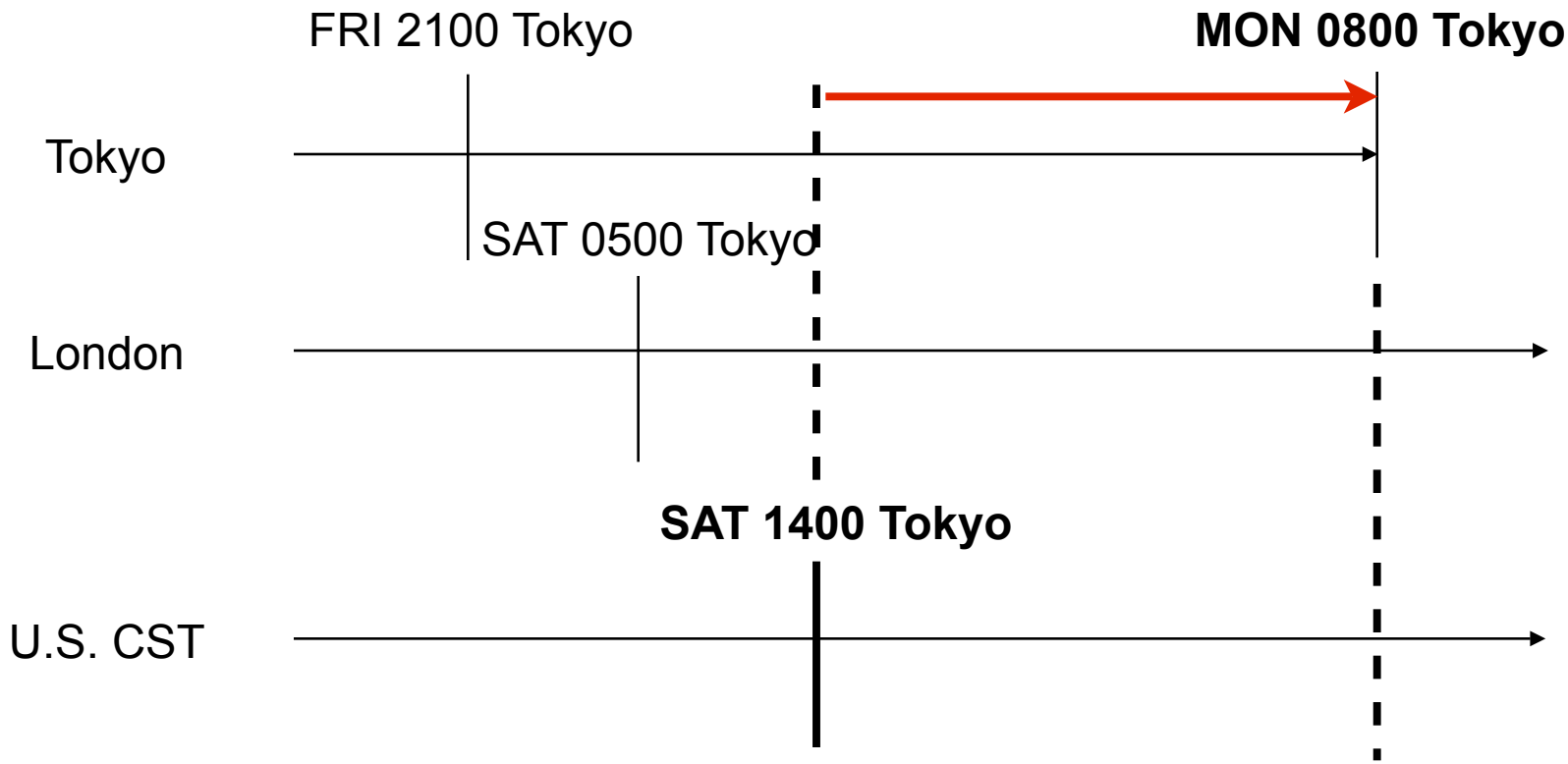
Global Operations - U.S. Perspective

must support u.s. west coast operations
all systems must be available until fri 2100 local time
must come back up mon 0800 local time



Global Operations - Tokyo Perspective

must support u.s. west coast operations
all systems must be available until fri 2100 local time
must come back up mon 0800 local time



how do you support the myriad of application updates and machine maintenance while still maintaining availability?

maintenance classification

type 1 updates

type 2 updates

type 3 updates

maintenance classification

type 1 updates type 2 updates type 3 updates

application or service-related updates that do not impact service contracts or require interface or data changes and simple administrative and configuration changes

maintenance classification

type 1 updates type 2 updates type 3 updates

simple bug fixes

changes to business logic (e.g., calculation)

changes to business rules

configuration file and simple administrative changes

supports active/passive cluster or active/active topology

maintenance classification

type 1 updates type 2 updates type 3 updates

application-related updates that require changes in interface contracts or service contracts in addition to other changes found in type 1 updates

maintenance classification

type 1 updates type 2 updates type 3 updates

additional user interface fields or screens

modifications to interfaces

modifications to service contracts

modifications to message structure

updates or fixes to XML schema definitions

requires the use of versioning in a HA/CA environment

supports active/passive cluster or active/active topology

maintenance classification

type 1 updates type 2 updates **type 3 updates**

updates that require coordination and synchronization of all components or updates involving shared memory or database schema changes

maintenance classification

type 1 updates type 2 updates type 3 updates

shared or local database schema changes

changes to objects located in shared memory

hardware upgrades and migrations

not supported through active/passive ha cluster

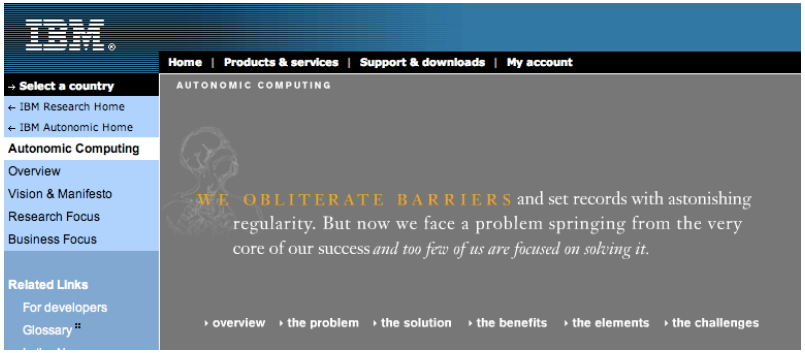
supports active/active ca topology

maintenance classification

why only three update types?

increased deployment complexity means increased risk of operator error, thereby affecting availability within the CA environment

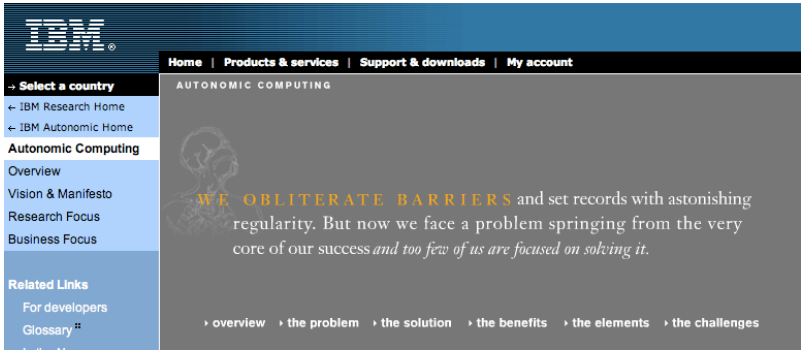
autonomic computing



http://www.research.ibm.com/autonomic/

a systemic view of computing modeled after a self-regulating biological system

autonomic computing



the vision: a network of self-healing computer systems that manage themselves

components that are self-configured

components that are self-healing of faults

components that are self-optimized to meet requirements

components that are self-protected to ward off threats

recovery-oriented computing

The Berkeley/Stanford Recovery-Oriented Computing (ROC) Project

http://roc.cs.berkeley.edu/

recovery-oriented computing focuses on recovering quickly from software faults and operator errors

recovery-oriented computing

based on the Peres rule, we need to cope with inevitable hardware and software failures

contain a fault in a component so it doesn't affect other components

automatically locate the root cause of the failure

repair the fault at the smallest subcomponent level

ability to inject faults for testing and training

detect and recover at the lowest possible level

Summary

References

- resource oriented computing: http://roc.cs.berkeley.edu/
- autonomic computing: http://www.research.ibm.com/autonomic/
- the availability digest: http://www.availabilitydigest.com