

An Active Approach to Characterizing Dynamic Dependencies for Problem Determination

Aaron Brown

Computer Science Division
University of California at Berkeley

Gautam Kar, Alexander Keller

IBM T.J. Watson Research Center

IM 2001, 16 May 2001

Slide 1

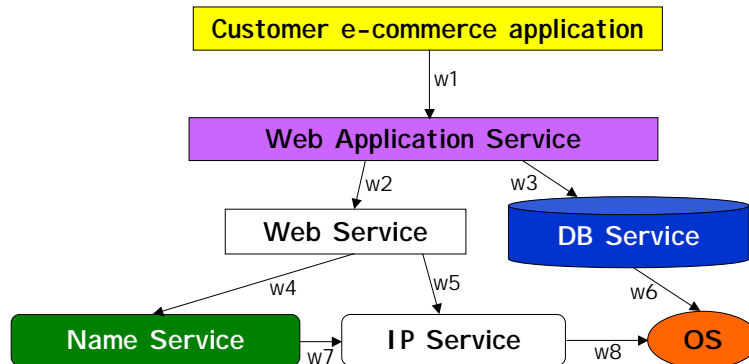
Motivation: problem diagnosis

- **Troubleshooting problems is one of the most challenging, time-consuming management tasks**
 - symptoms are typically at end-user or SLA level
 - root causes are typically much deeper in system
 - » and often confounded by system complexity
 - **must map symptoms to root causes to locate problems!**
- ***Dependency models* provide an invaluable aid to root-cause analysis**
 - capture connections between high- and low-level system components

Slide 2

Dependency models in a nutshell

- Use a graph (DAG) structure to capture dependencies between system components
 - if failure of **A** affects **B**, then **B** depends on **A**
 - edge weights represent dependency strengths



Slide 3

Constructing dependency models

- For effective diagnosis, model must capture:
 - static dependencies
 - dynamic runtime dependencies
 - » *e.g.*, dependencies induced by runtime queries
 - distributed dependencies
 - dependency strengths
 - all at the detailed level of individual system resources
- Most existing techniques don't meet these challenges...

Slide 4

Outline

- Motivation & background
- **ADD: Active Dependency Discovery**
- Experimental validation of ADD
- Conclusions and future directions

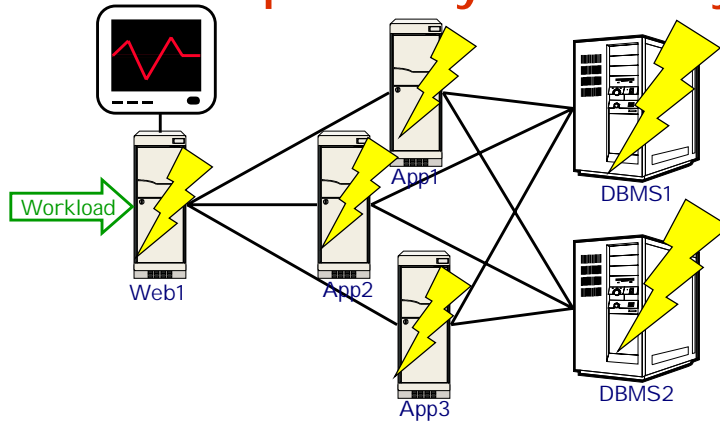
Slide 5

Discovering dependencies

- **Desired properties of approach**
 - identifies dynamic, runtime dependencies
 - works on distributed systems
 - works with only black-box view of system components
 - provides direct evidence of causality
 - detects dependencies only visible in failure situations
- **These properties inspire an indirect, active approach**
 - **indirect**: no explicit modeling of system
 - **active**: perturb system to elucidate dependencies

Slide 6

Active Dependency Discovery (ADD)



- 1) Instrument the system and apply workload
- 2) Systematically perturb components
- 3) Measure change in system response
- 4) Construct dependency model from measurement data

Slide 7

Benefits of ADD

- **Coverage**
 - no need to rely on problems occurring naturally, as in passive approaches
 - active perturbation explicitly elucidates dependencies between all system components
 - » if perturbation is realistic
- **Causality**
 - causality easy to establish: perturbation is the cause
- **Simplicity**
 - no application modeling or modification necessary
 - existing SLA instrumentation may be sufficient
 - no complex data mining required
 - applied *before* real problems occur

Slide 8

Drawbacks of ADD

- **Invasiveness**

- can be tricky to do perturbation on production system
- possible solutions:
 - » leverage redundancy if available (*e.g.*, cluster system)
 - » run perturbation during non-production periods (initial system setup or during scheduled downtime)
 - » develop low-grade perturbation techniques

- **Workload-specific**

- extracted models only valid for applied workload
- but, can model components of workload and recombine later

Slide 9

Outline

- Motivation & background
- ADD: Active Dependency Discovery
- **Experimental validation of ADD**
 - approach
 - TPC-W testbed environment
 - results
- Conclusions and future directions

Slide 10

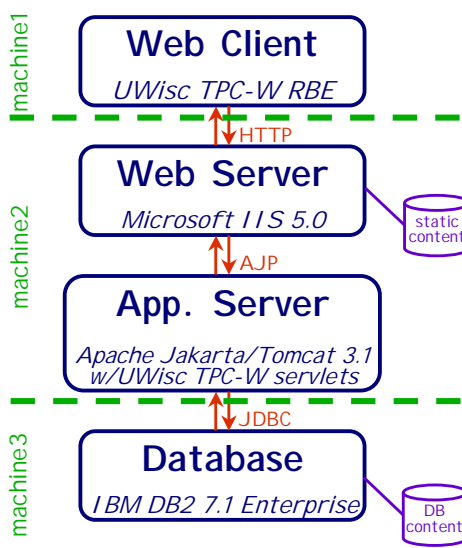
Validation: e-commerce case study

- **Goal: use ADD to discover dependencies in a multi-tier e-commerce environment**
 - using off-the-shelf black-box software
 - in a realistic environment with realistic workload
- **Task: discover dependencies of user web requests on database tables**
 - for each type of user request:
 - » extract dependencies on individual database tables
 - » characterize strengths of those dependencies
 - » hand-verify model against application source code
- **Platform: TPC-W benchmark app & workload**
 - realistic mockup of online bookseller e-commerce site

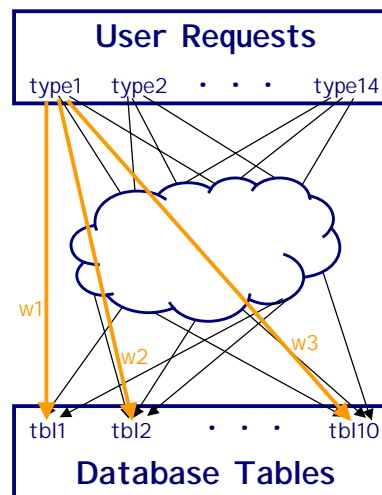
Slide 11

TPC-W experimental testbed

System View



Dependency View



Slide 12

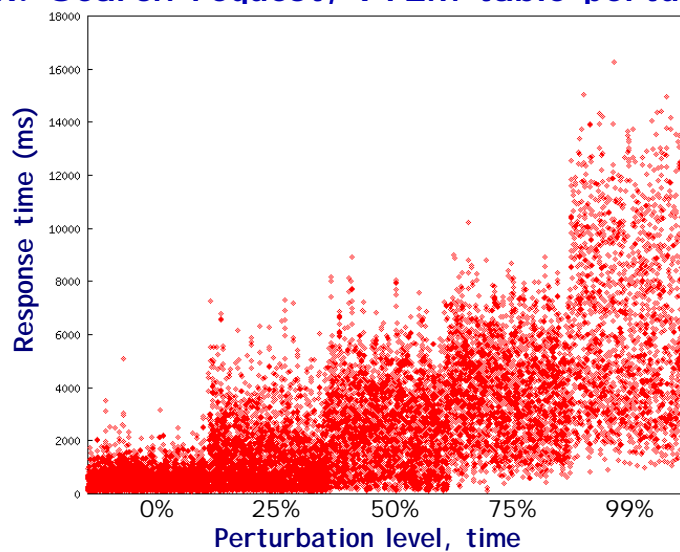
Perturbation and measurement

- **Perturbation applied to individual DB tables**
 - use DB2's lock manager to exclusive-lock a table
 - configurable "duty cycle" of lock out
 - » queries locked out for first x% of every 4 sec. interval
 - only affects one table; no impact on overall load
 - can simultaneously perturb multiple tables
- **Per-request response time measured by TPC-W front-end user emulator**
 - 14 different types/classes of requests
 - response time is end-to-end, including network delay

Slide 13

Raw perturbation results

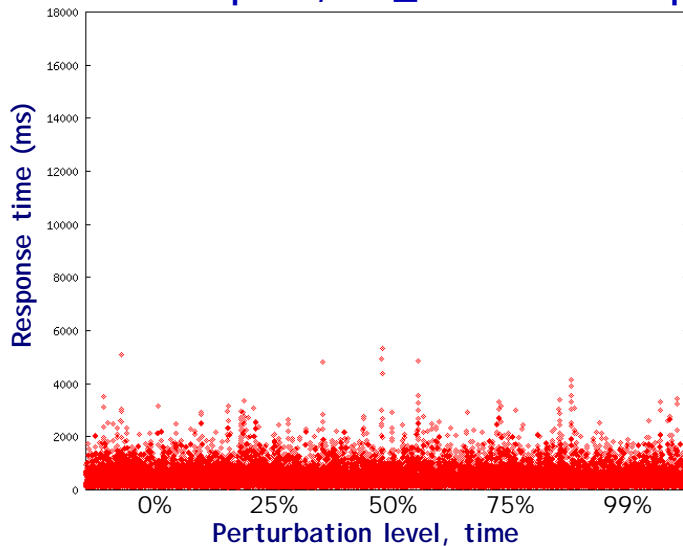
- **Ex: Search request, ITEM table perturbed**



Slide 14

Raw perturbation results (2)

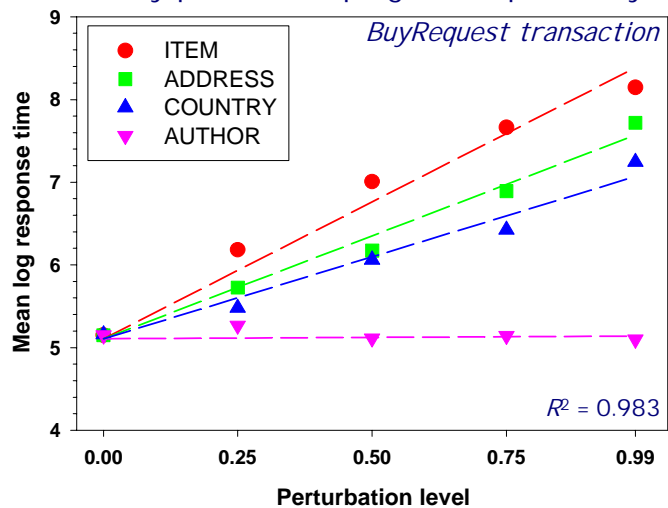
- Ex: Search request, CC_XACTS table perturbed



Slide 15

Applying a linear model

- Linear regression on mean of log of data
 - statistically positive slope gives dependency strength



Slide 16

Summary of results

- Modeling correctly identified 41 of 42 true dependencies at 95% confidence level
 - compare to 140 potential dependencies (!)
 - one false negative most likely due to insufficient data
 - many non-dependencies had statistically significant negative effects/strengths
 - » due to an interaction not captured by model
 - » solution: improve model or simply discard negative strengths

Slide 17

Summary of results (2)

- Tabular representation of full dependency set:

Request Table	admconf	admreq	bestsell	buyconf	buyreq	custreg	home	newprod	orddisp	ordering	proddet	srchreq	srchres	shopcart
ADDRESS				X	X				X					
AUTHOR	X	X	X					X			X		X	
CC_XACTS				X					X					
COUNTRY				X	X				X					
CUSTOMER				X	X	X			X					
ITEM	X	X	X	X	X		X	X	X		X	X	X	X
ORDER_LINE	-		X	X					X					
ORDERS	X		X	X					X					
SHOP_CART														X
SHOP_CART_L				X	X									X

Strengths: X = (0,1] X = (1,2] X = (2,3] X = (3,4]

Slide 18

Using dependencies for diagnosis

- **When a problem occurs:**
 - 1) identify faulty request
 - » from problem report, SLA violation, test requests, ...
 - 2) select the appropriate column in dependency table
 - 3) select the rows representing dependencies
 - » this is the set of potential root causes
 - 4) investigate potential root causes, starting with those of highest weight

Slide 19

Using dependencies for diagnosis (2)

- **Can extend approach to multiple system levels**
 - compute one dependency matrix per level
 - iterate levels from user symptoms to culprit resource
- **This process may not uniquely identify problem**
 - but can narrow down the culprits via combinations
 - » isolating the effects of individual tables
 - » e.g., SHOP_CART_L "=" orderdisp - buyconf
 - not all tables can be uniquely isolated
 - » but could do so by adding synthetic test requests?
 - » ideal is to build a *basis* for the whole-system dependency matrix

Slide 20

Outline

- Motivation & background
- ADD: Active Dependency Discovery
- Experimental validation of ADD
- **Conclusions and future directions**

Slide 21

Conclusions and future directions

- **Dependency models help problem determination**
- **ADD effectively discovers dependency models**
 - approach is uniquely positioned in the design space
 - » active, indirect approach finds dynamic, distributed dependencies; works on black-box systems
 - initial experimental results are promising
 - » very good success on TPC-W experiments
- **Future directions**
 - techniques to integrate ADD into production systems
 - investigation of end-to-end vs. layer-by-layer tradeoffs
 - using dependency models for other management tasks
 - » impact analysis, performance optimization, ...

Slide 22

An Active Approach to Characterizing Dynamic Dependencies for Problem Determination

For more information:

abrown@cs.berkeley.edu

{gkar,alexk}@us.ibm.com

<http://www.research.ibm.com/sysman>

Slide 23