Confirming and Reconfirming Architectural Decisions: A Goal-oriented Simulation Approach

A Dissertation Presentation

By

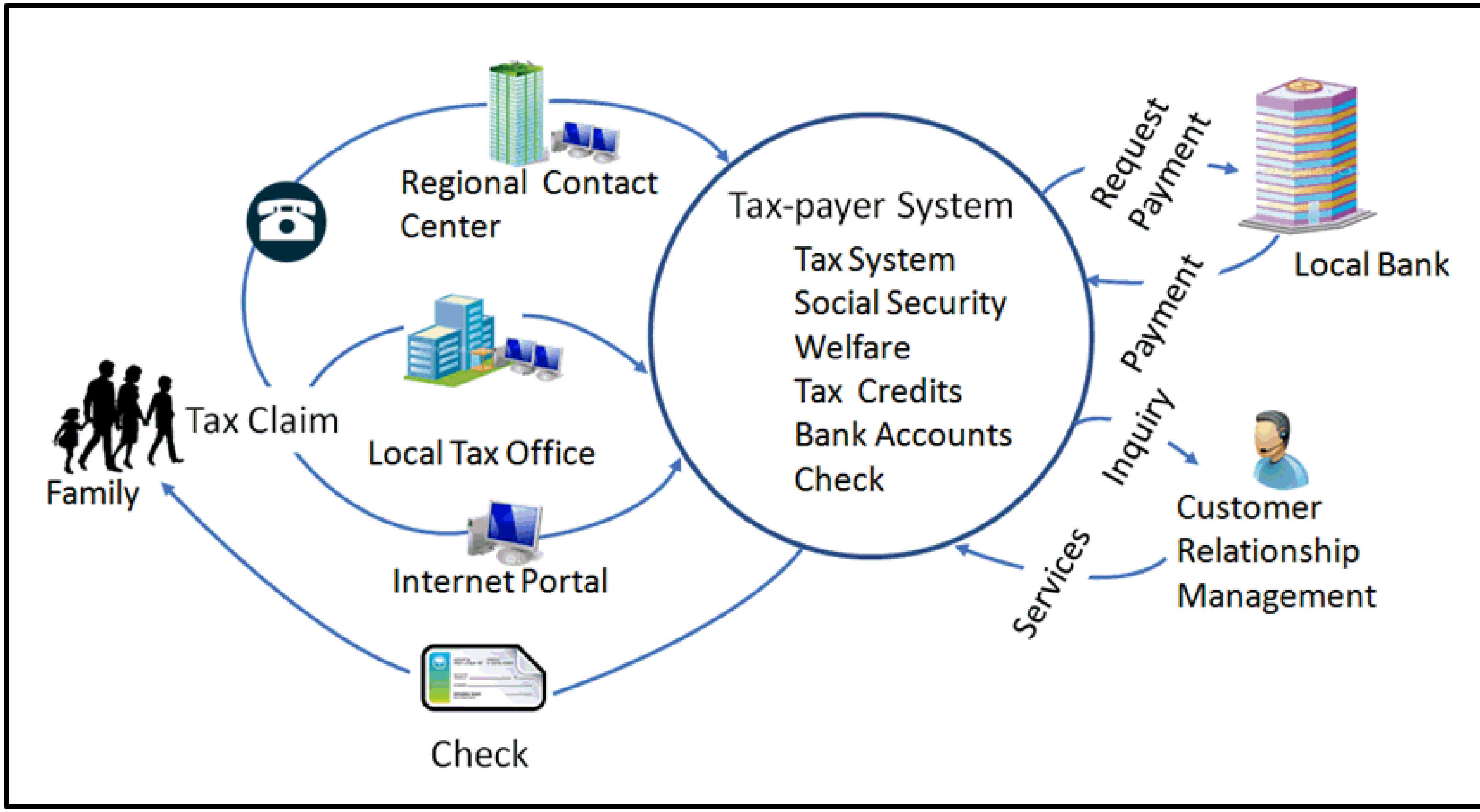
Thomas L. Hill

Ph.D. Supervisory Committee
Dr. Lawrence Chung (Chair)
Dr. Farokh Bastani
Dr. Kang Zhang
Dr. Latifur Khan
Dr. Eric Wong

Outline

- Motivation
- Research Problem
- Related Work
- The Proposed Solution
  - GoBench
  - GoSim
- Case Studies
- Conclusion

"Evaluate a poor performing complex national tax system"



- Software is functionally sound
- Architecture fails the goals of users due to:
  - Poor performance
  - Unexpected development and maintenance cost
- Engineers unable to predict or confirm architecture behavior

Motivation

- The architecture evaluation points of interest:
  - Design was not confirmed or reconfirmed via analysis, benchmarking, simulation or volume testing
  - Design confirmed when system placed into production
  - Non-functional requirements defined in a 200 page service level agreement
  - The behavior of the system was too complex to understand or maintain
  - The application workload was not documented or matched to the performance characteristics of the run-time infrastructure

Outline

- Motivation
- Research Problem
- Related Work
- The Proposed Solution
    - GoBench
    - GoSim
- Case Studies
- Conclusion

Research Problem

- **Today's software engineers are unable to assess or predict a system-architecture's ability to satisfy stakeholder performance and cost goals, in a fast and inexpensive manner.**
- **The tools required to quickly understand, assess and predict the behavior of complex cloud architectures are disconnected and in a nascent stage of usage by industry software engineers.**

Unanswered Questions?

Research Problem - Specific

- Several unanswered questions remain as obstacles in the path to understand the behavior of these modern systems:
  1. Why are stakeholder NFR-goals expressed as natural language contract-binding service level agreements?
  2. Where are the online transaction processing (OLTP) benchmarks results for cloud architectures?
  3. How can the limits of cloud architecture resource elasticity be discovered?
  4. Why is it so difficult to describe a discrete event simulation model experiment?
  5. What basic software engineering artifacts and tools are needed to understand the behavior of a complex enterprise-level system throughout its development and operational life?

Research Goal and Approach

Research Goal: This research seeks to make a difference throughout the software development and maintenance lifecycle by using benchmarking and new discrete event simulation modeling techniques to integrate: NFR goals, workload and architecture infrastructure.

Approach:

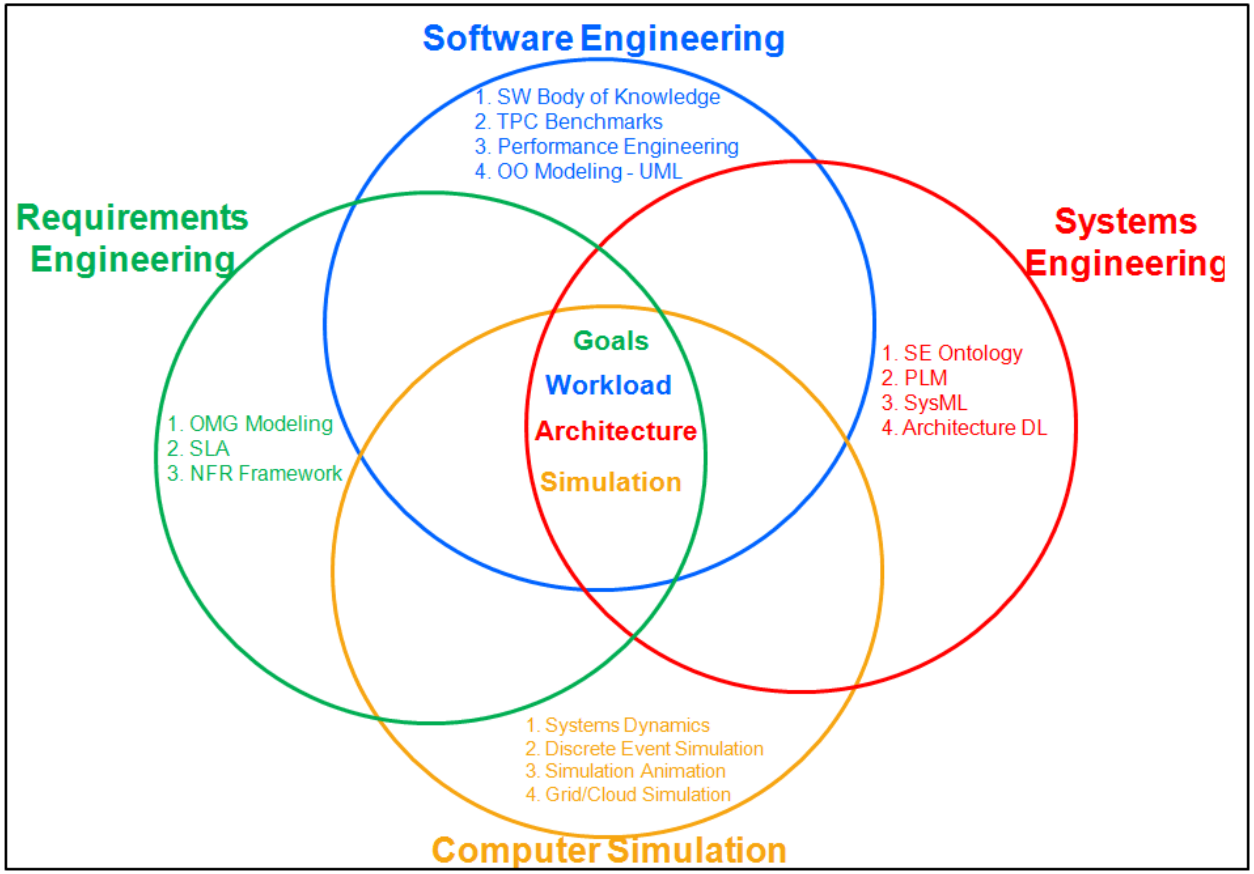
- **Build on NFR goal graphical representations as softgoals**
- **Use standard benchmarking to specify performance goals, requirements, database definitions and transaction workload characteristics**
- **Generate multiple benchmark experiments to collect actual performance and resource usage of multiple architectures**
- **Use multiple open source discrete event simulators to model the benchmarked performance goals, requirements, database definitions and transaction workload characteristics**
- **Compare benchmark results to simulation results to authenticate the fidelity of simulation as an architecture reconfirming tool**

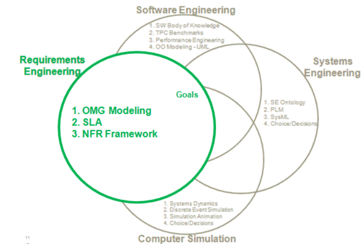
Outline

- Motivation
- Research Problem
- Related Work
- The Proposed Solution
  - GoBench
  - GoSim
- Case Studies
- Conclusion

Related Work

- Intersection of four evolving engineering domains:
  - Requirements Engineering
  - Software Engineering
  - Systems Engineering
  - Computer Simulation



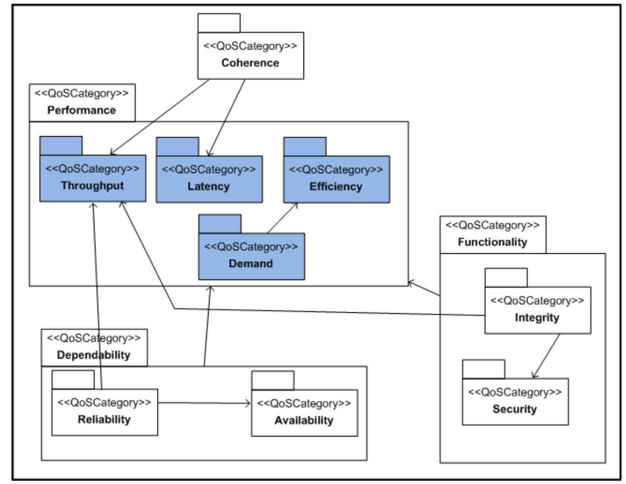


Related Work

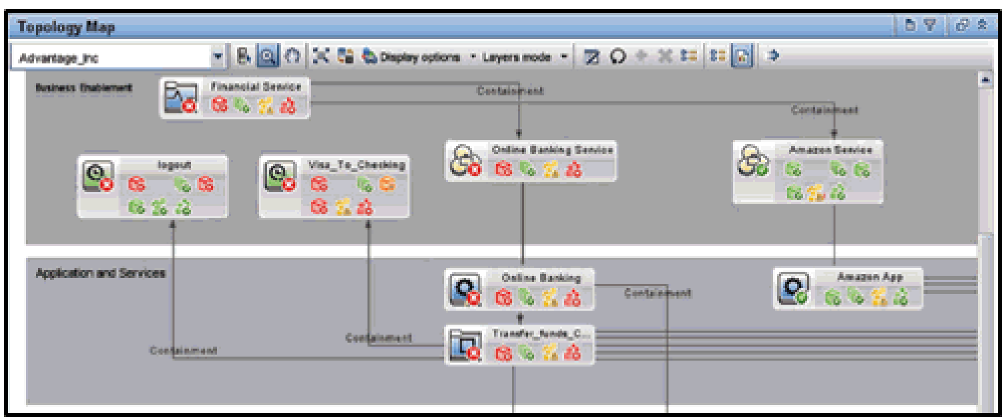
Requirements Engineering

UML Profile for Modeling QoS [OMG06]

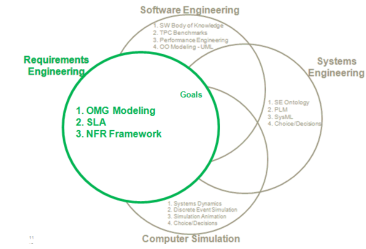
+ UML extension specifications through stereotypes
+ System concerns: User satisfaction and resource consumption
+ Categories (performance, dependability, security, integrity, coherence, throughput, latency, efficiency, demand, reliability, availability)
- Definitions only; system goals, design, implementation missing



Service Level Agreements and Monitoring [EDS/HP10]



+ Service Level Agreement (SLA) is a contracted system performance goal
+ SLA components (what provider promises, how delivered, who will measure, what penalties provider will pay)
+ HP Transaction Summary monitoring display
- Not traceable to system design requirements



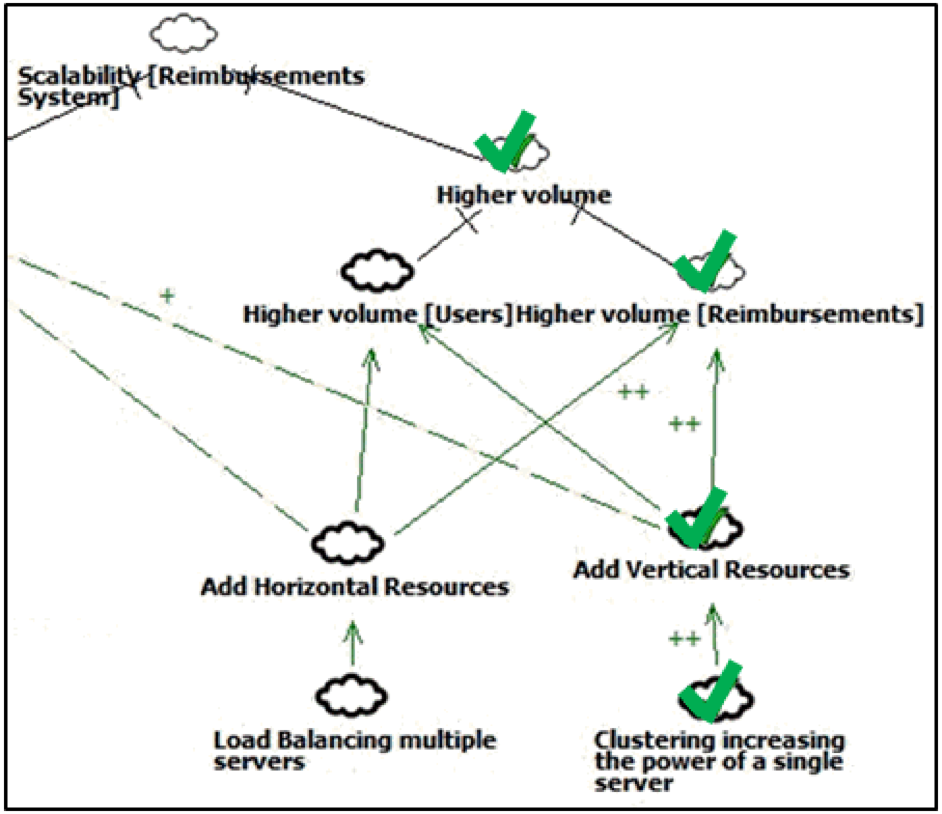
Related Work

Requirements Engineering

NFR Framework - Goals

a. NFR in Software Engineering [CNYM00]

b. Confirming and Reconfirming Architectural Decisions on Scalability: A Goal-Driven Simulation Approach [HSC09]



+ Non-functional requirements represented as softgoals (Softgoal Interdependency Graph)

+ Goal oriented analysis, document decisions rationale

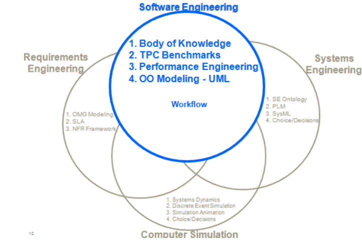
+ Simulation to assist making architecture decisions

- No integration of goals, transaction flow and architecture

- Only scalability goal researched

Related Work

Software Engineering

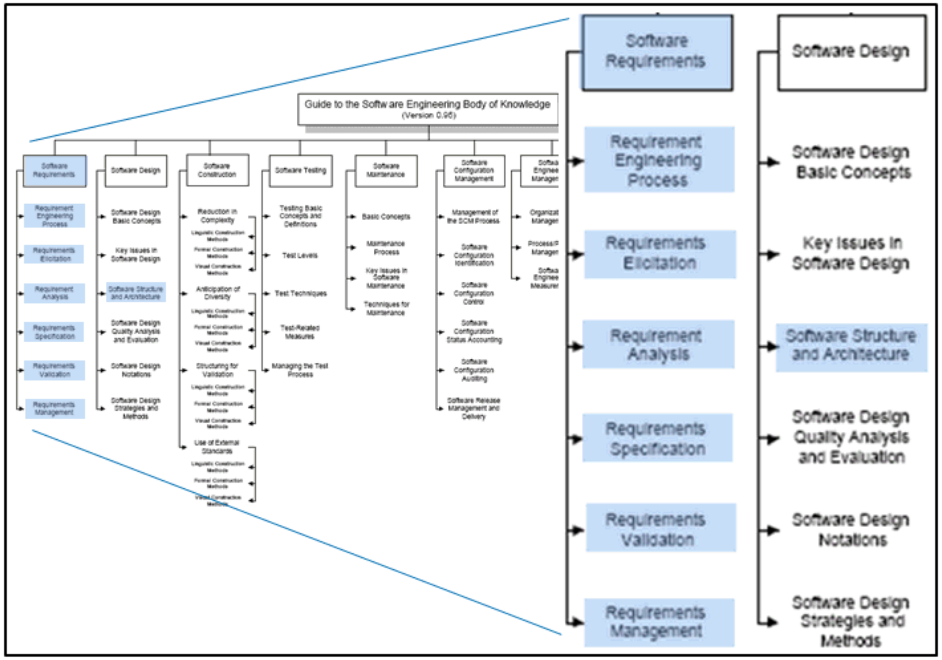


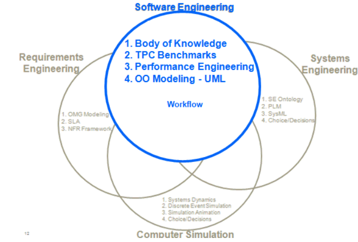
Software Engineering Body of Knowledge [IEEE04]

+ Computer scientists extend knowledge, software engineers build artifacts

+ 10 key knowledge areas 14 deep (requirements to quality)

- Concerned with process and lifecycle; goals not mentioned





Related Work

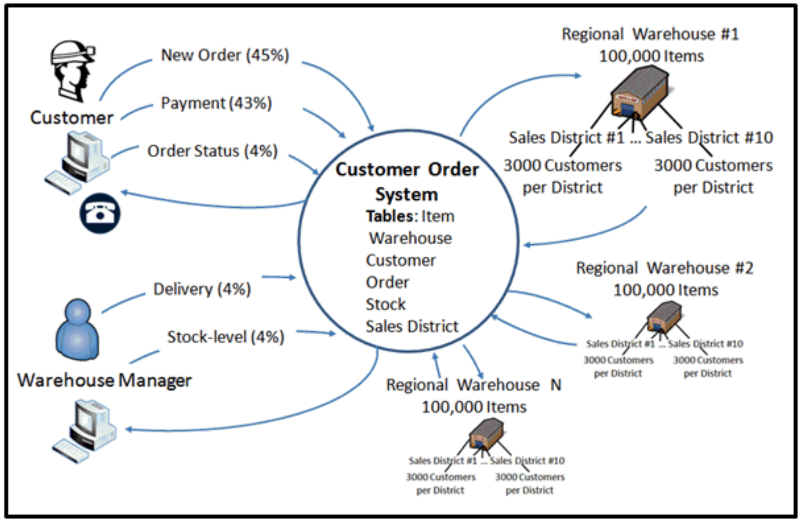
Software Engineering

Transaction Processing Benchmarks [TPC-C11]

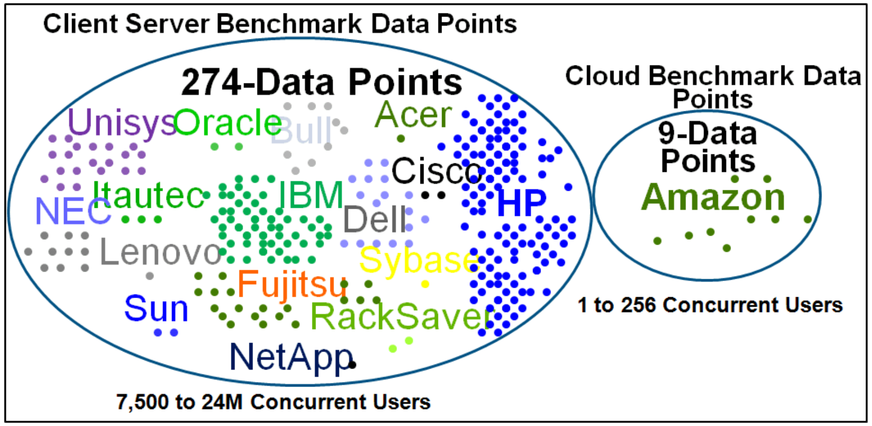
+ Standard objective verifiable performance and cost OLTP, RDB since 1992
+ Business throughput metrics; number of orders processed per minute with cost

- OLTP and relational database only

- High cost to benchmark, high cost to customize



Transaction Processing Council Benchmarks [TPC-C13]

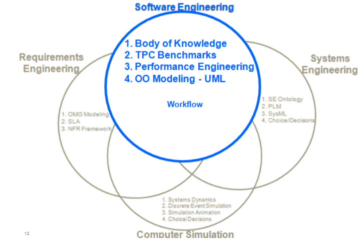


+ 274 client server benchmarks documented
+ 9 cloud benchmarks using Amazon cloud created by Stony Brook University

- No cloud benchmarks for Google, Microsoft, HP

Related Work

Software Engineering

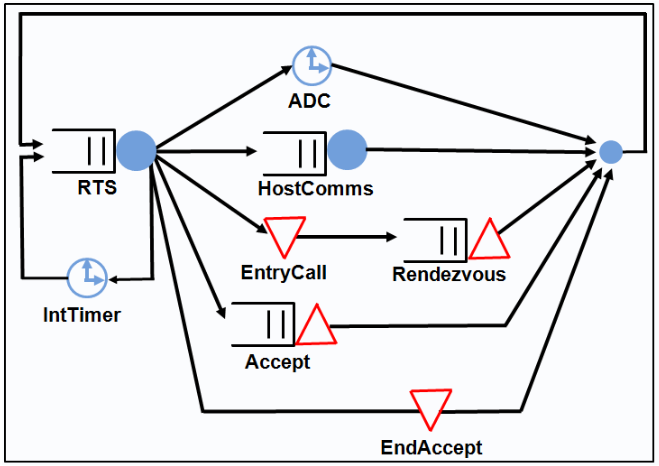


Software Performance Engineering [Smith93]

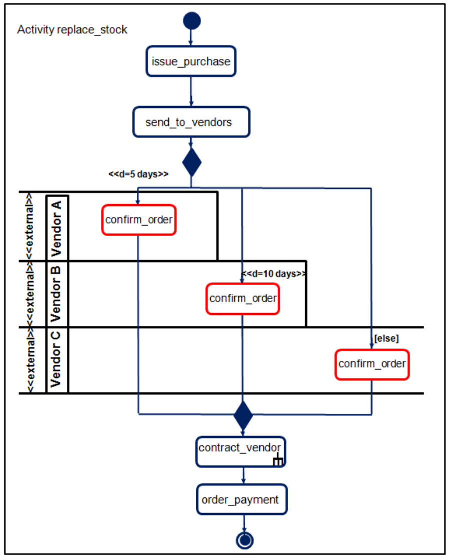
+ Analysis strategies (adapt-to-precision, simple-to-realistic, best-and-worst-case)

+ SPE data (performance requirements, behavior patterns, software description, execution environment, resource usage estimates)

- Petri net model analysis training needed



Extending and Formalizing UML 2.0 Activity Diagrams … [Chung10]

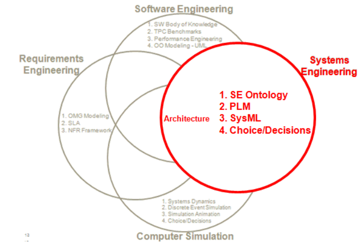


+ UML Activity diagrams can be used to document the workflow of business and computer functions

- Need to extend overlay of goals on workflows and architecture

Related Work

Systems Engineering



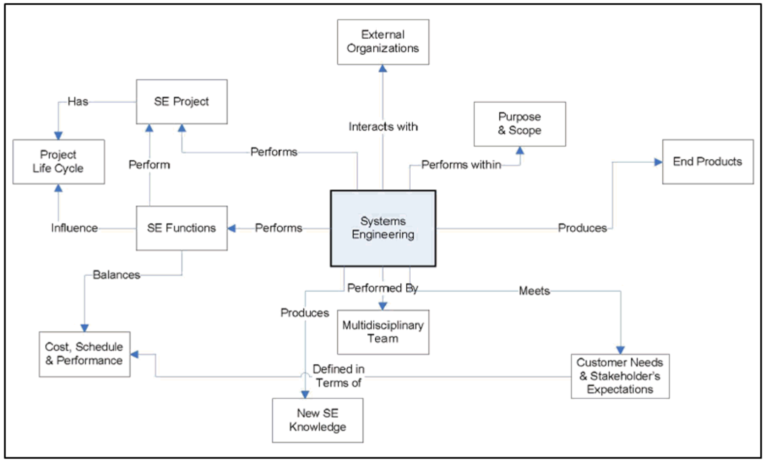
Developing Systems Engineering Ontology [Sarder07]

+ Taxonomy of systems engineering functions

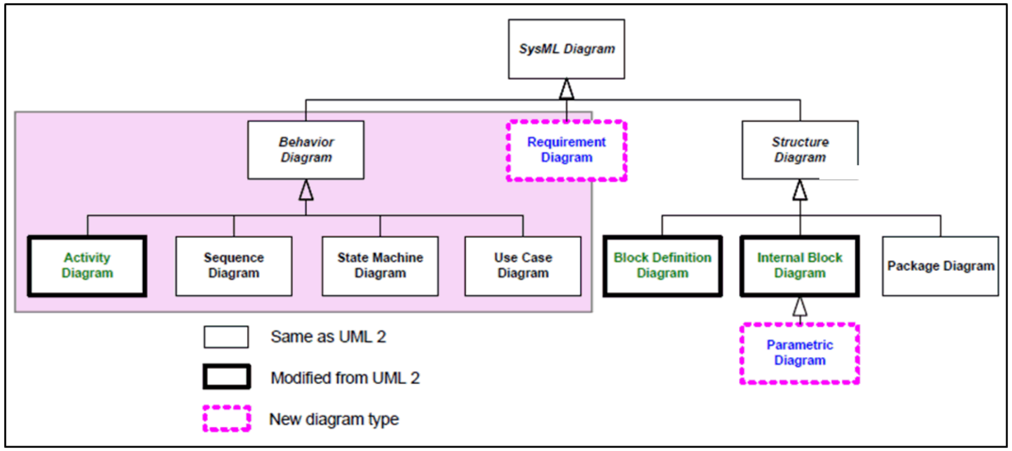
+ Top level systems engineering ontology

- Only high-level definitions provided

- Early stage, design not complete



SysML/UML 2 Behavior Diagrams – Systems Engineering Handbook [INCOSE11]



+ SE lifecycle detail definition

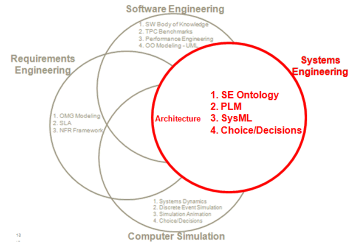
+ Practice of architecture design (SysML-OMG-INCOSE, DODAF, MODAF)

+ Modeling, simulation, prototyping defined

- Little mention of goal-orientation

Related Work

Systems Engineering



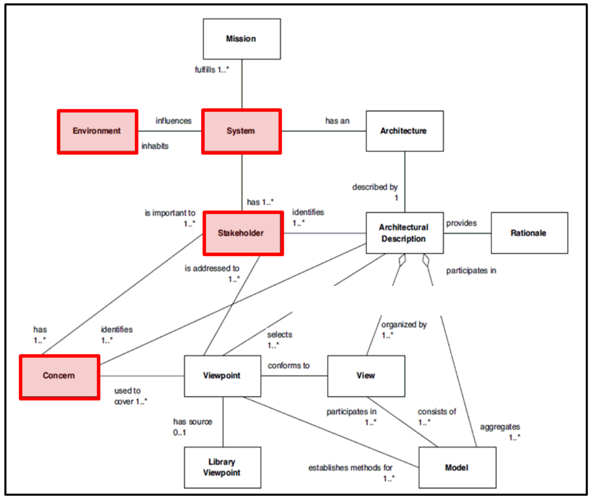
ISO/IEC 42010 Systems and Software Engineering – Recommended Practice for Architecture Description of Software-intensive Systems [IEEE07]

+ Expression and communication of architecture

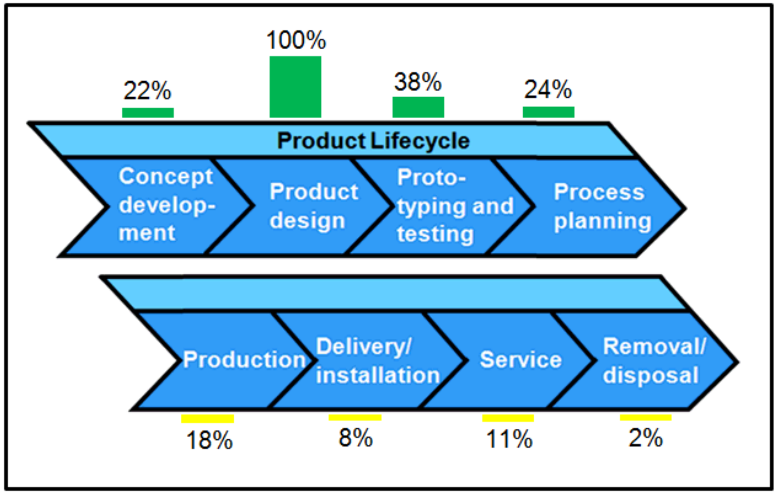
+ Conceptual model of an Architectural Description (AD)

- Deals with what, no elaboration of how to develop an AD

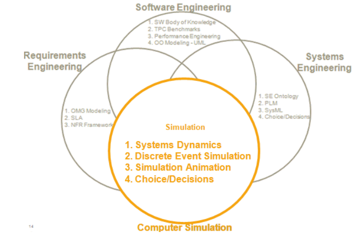
- No mention of how to analyze an AD



PLM-CAD/CAM, BoM, Simulation [Siemens11]



+ UGS product lifecycle visualization Teamcenter digital prototyping and plant simulation
+ Lifecycle concept-development to removal-disposal
+ CAD, CAE, CAM, Digital Manufacturing, FEA, PDM
- Expensive overkill for modeling information technology systems



Related Work

Computer Simulation

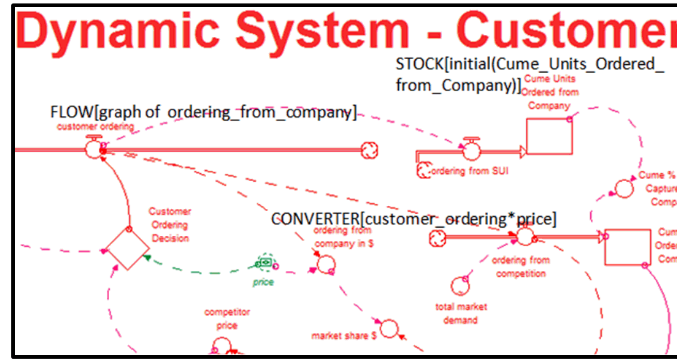
System Dynamics - Industrial Dynamics [Forrester61]

+ The noteworthy beginnings of management as a science and systems dynamics

+ Building experimental models of companies and industries –DYNAMO compiler

+ Stock and flow simulation predecessor

- No integration of goals, workflow and infrastructure



Discrete Event Simulation – Simulation Modeling and Analysis [Law91]

Discrete Event Simulation M/M/1 Algorithm

```
procedure mm1 (Compressed from mm1.c, Law91)
Input: meanInterarrivalTime seconds and meanServiceTime seconds
Output: performance metrics; average delay in queue, average number in queue, server utilization
Initialize simClock =0, nextEvent = simClock + exponentialFunction(meanInterarrivalTime)
while endCondition  false do
   Determine  nextEventType
   if eventList empty then
      endCondition = true
   end
   Advance simulation clock simClock = minimumTimeNextEvent
  Accumulate performance metrics
  if nextEventType arrival then
     Schedule next arrival nextEvent = simClock + exponentialFunction(meanInterarrivalTime)
     if serverBusy then
        Accumulate performance metrics
     else
        Schedule a departure nextEvent = simClock + exponentialFunction(meanServiceTime)
     end
  else nextEventType is depart
     if queueEmpty then
        serverBusy = false
     else
        Schedule a departure nextEvent = simClock + exponentialFunction(meanServiceTime)
     end
  end
end
Produce performance metrics report
```

+ The teaching “Bible” of Discrete Event Simulation (DES) since 1982

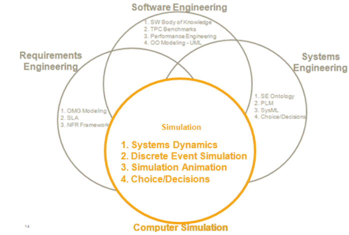
+ Basic components of DES model of a system that changes over time (state, clock, event list, timer)

+ Simple modeling icons and simulation program samples in Fortran and C

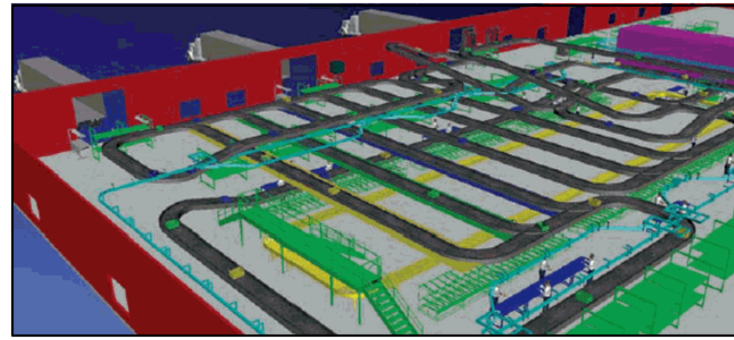
- No integration of goals, workflow and infrastructure

Related Work

Computer Simulation

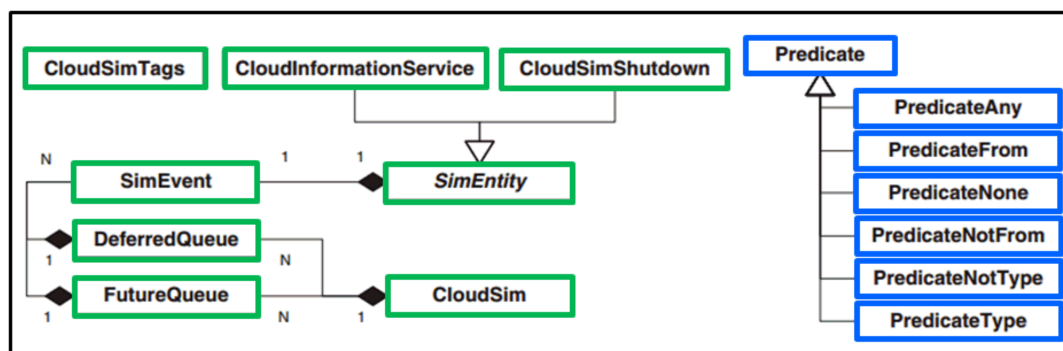


Simulation animation [Siemens10]



+ Uses 3D CAD structures to visualize and animate plant flow
+ Operations animation to visualize performance
+ Communicate design alternatives to management
- Costly for IT systems

CloudSim: a toolkit for modeling and simulation of cloud computing environments .. [Calheiros10]



+ Derived from an operational grid simulator
+ Simulates using cloud components (datacenter, brokers, host, broker, VM, Cloudlets)
- No goal properties are considered
- Java used to modify workload and infrastructure variables

Outline

- Motivation
- Research Problem
- Related Work
- The Proposed Solution
  - GoBench
  - GoSim
- Case Studies
- Conclusion

Software Engineering Framework

GoBench GoSim Framework Steps Annotated with Artifacts

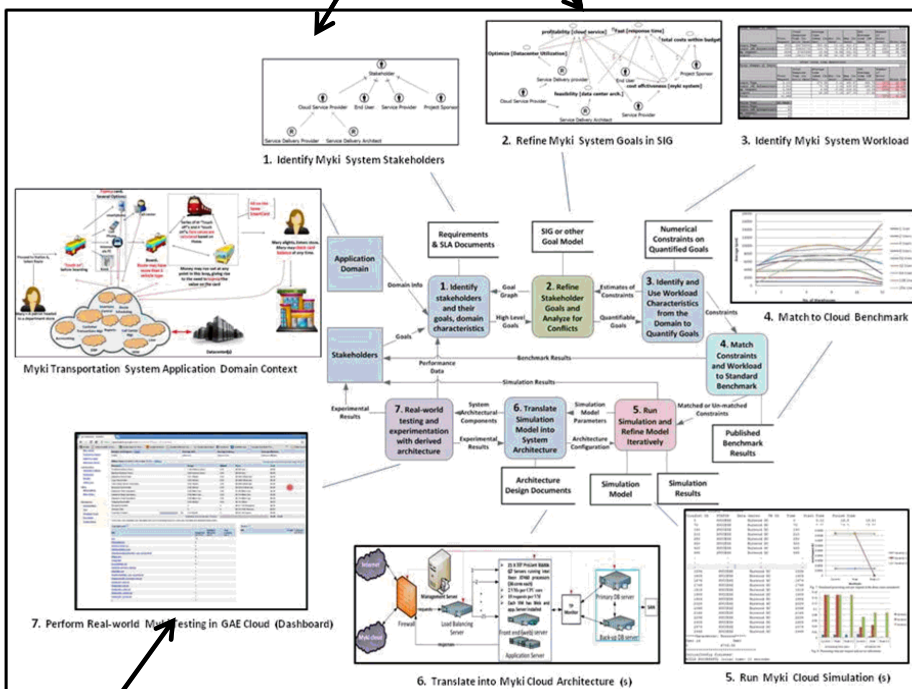
Non-functional requirements documented as Softgoal Interdependency Graphs

Estimation of application workload

Software application context diagram

Benchmark results of throughput and number of users

Simulation results of throughput , number of users and cloud datacenter



LoadRunner test cases and performance

Architecture implementation diagram

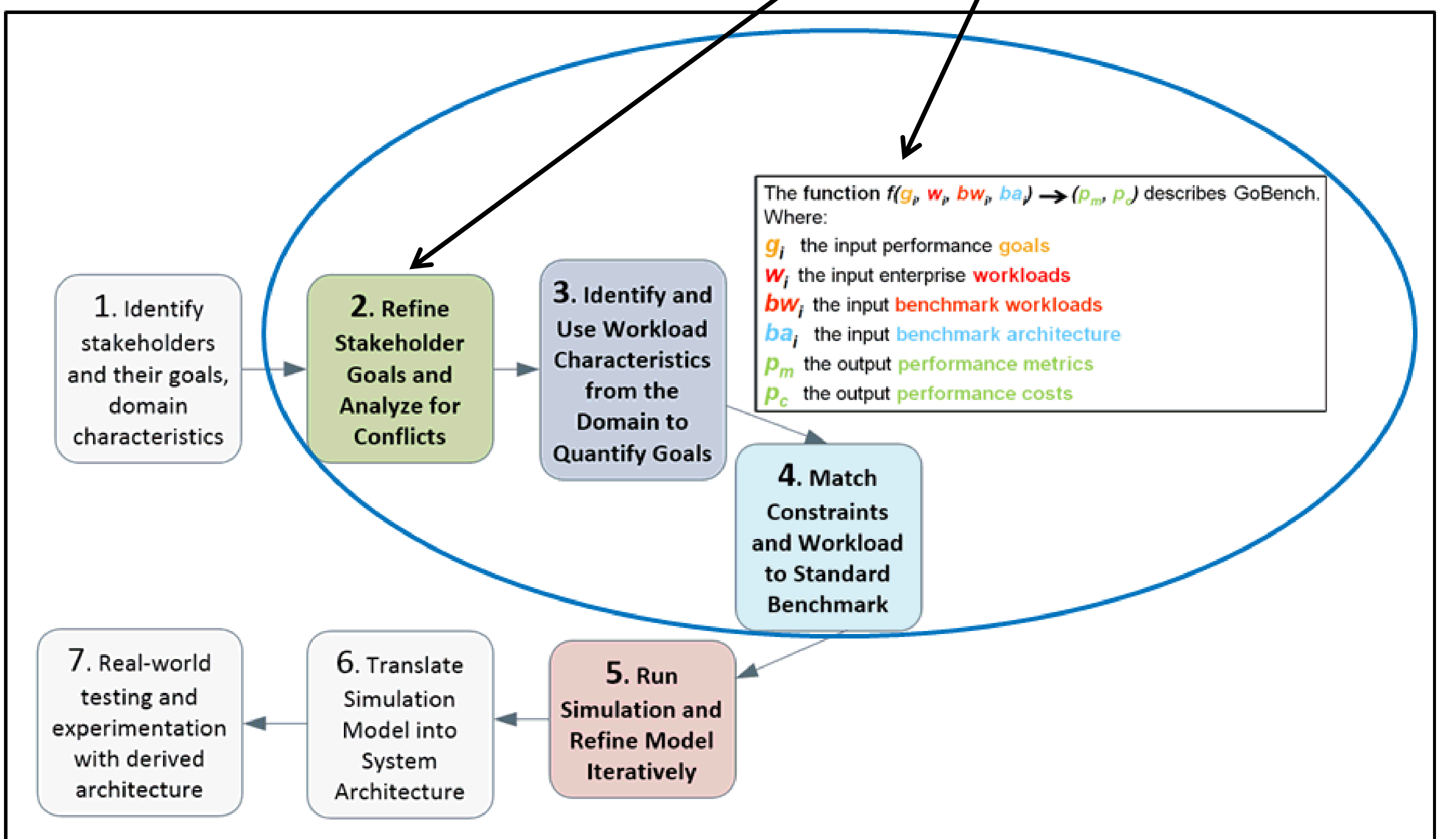
Outline

- Motivation
- Research Problem
- Related Work
- The Proposed Solution
  - GoBench
  - GoSim
- Case Studies
- Conclusion

GoBench Software Engineering Framework

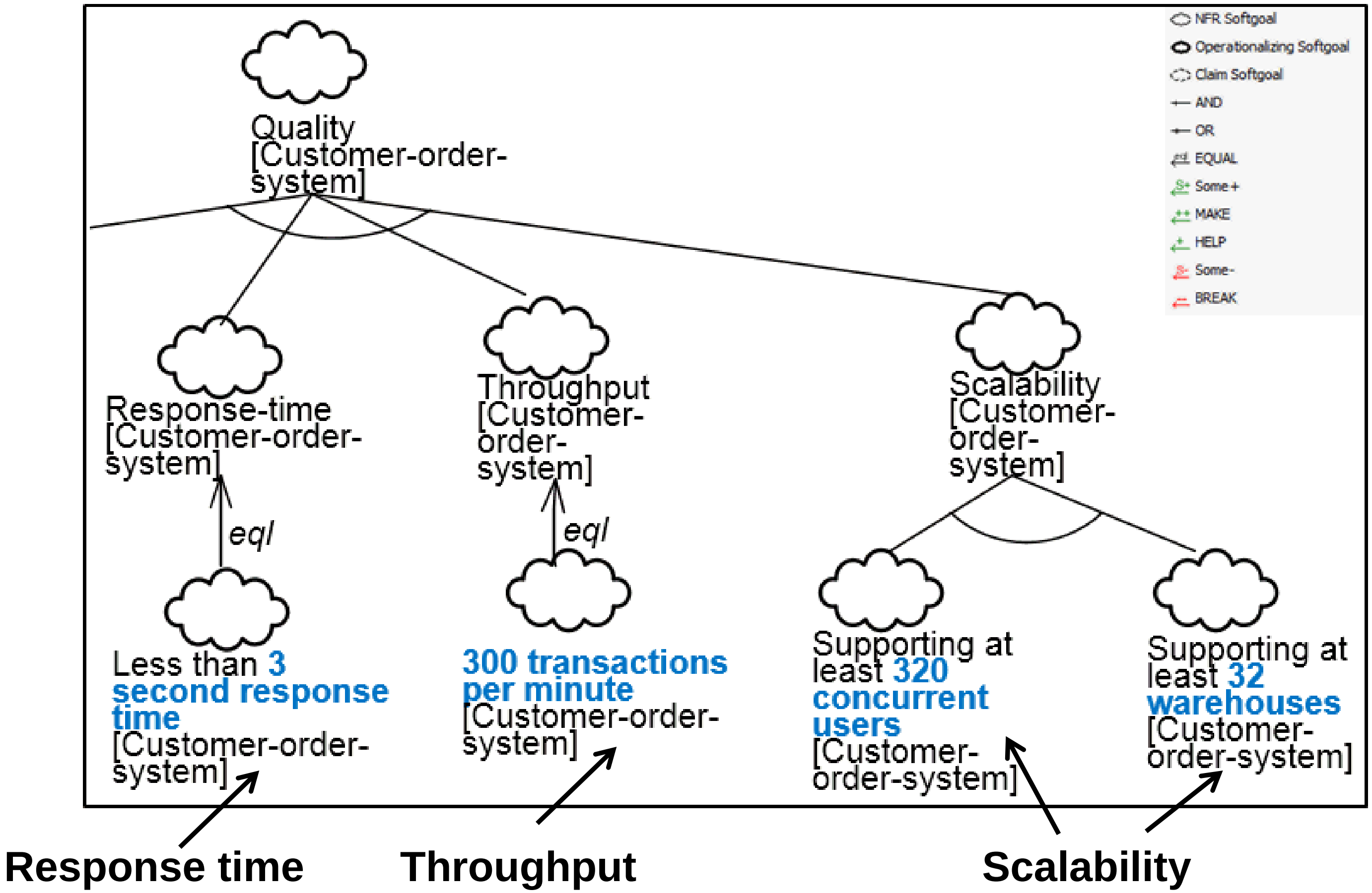
Seven Software Engineering Steps - Confirming

Step 2 (Stakeholder Goals) and Step 4 (GoBench Benchmark Matching Function) Highlighted



GoBench Softgoal Interdependency Graph

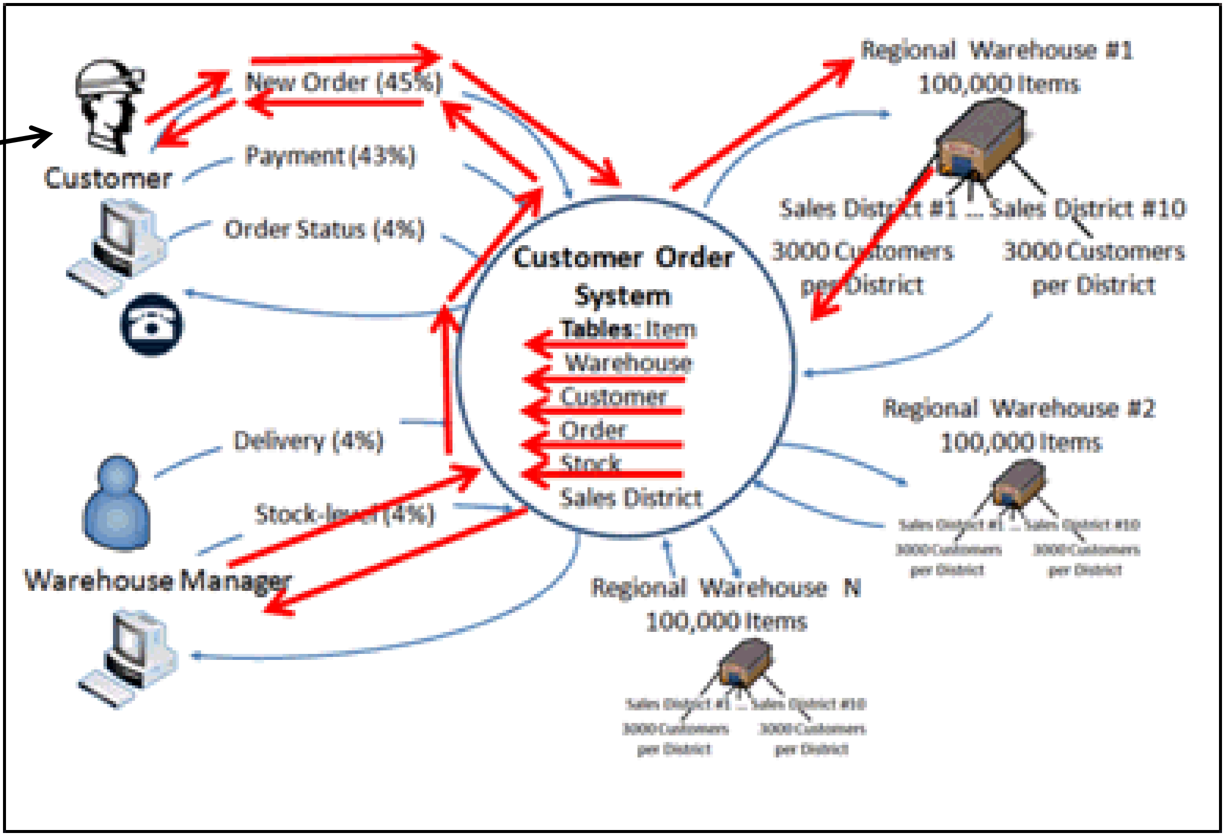
Step 2 Non-Functional Requirements Performance Goals



GoBench TPC-C Benchmark Context

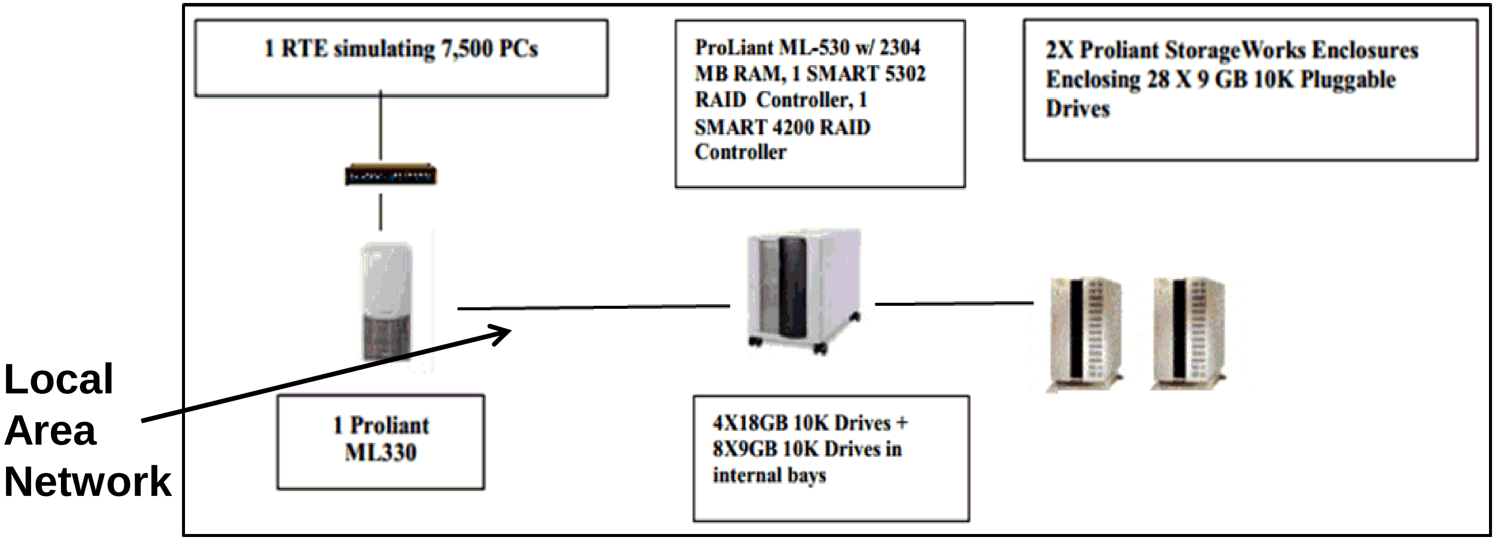
Step 4 Benchmark Application Workload and Flow

Customer Order transaction flow

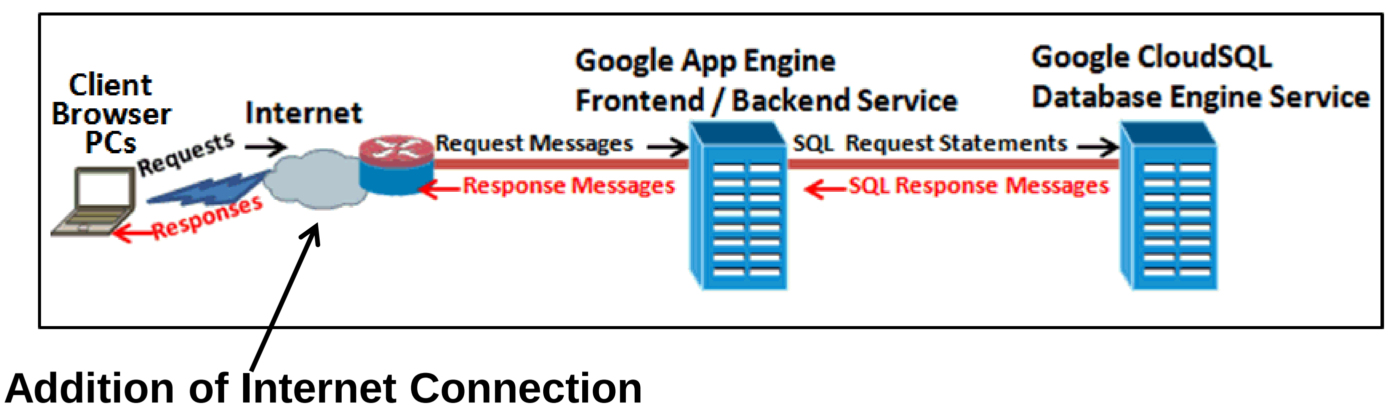


GoBench TPC-C Benchmark Architectures

Step 4 Benchmark Client Server Architecture



Step 4 Benchmark Cloud Services Architecture

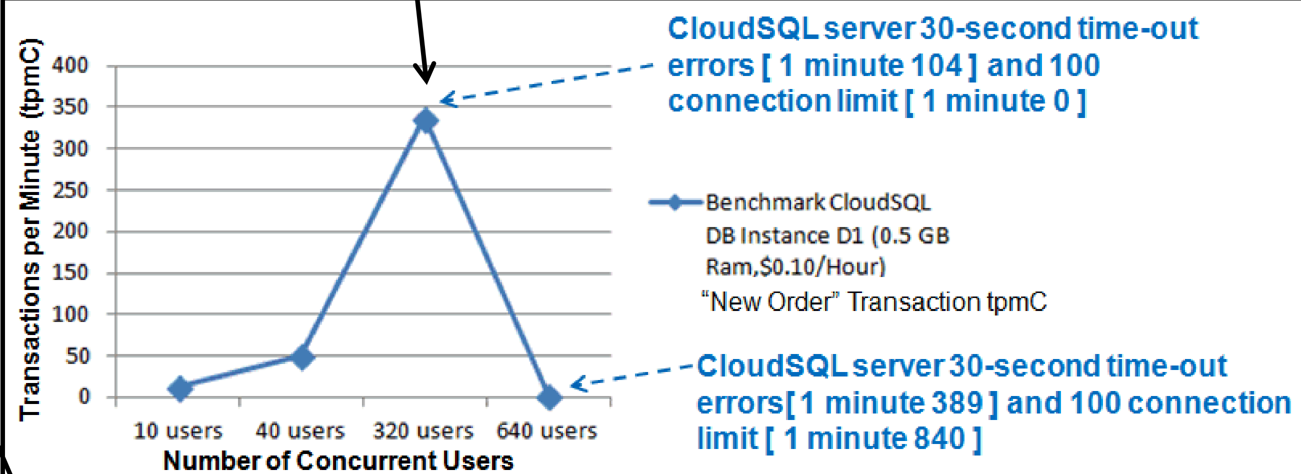


GoBench TPC-C Benchmark Results

Step 4 Google Cloud Results for a D1 CloudSQL Database Instance

The performance throughput knee of the D1 CloudSQL server (338.1 transactions per minute with 320 concurrent users)

Benchmark throughput in transactions per minute (tpmC) for a variable number of concurrent users



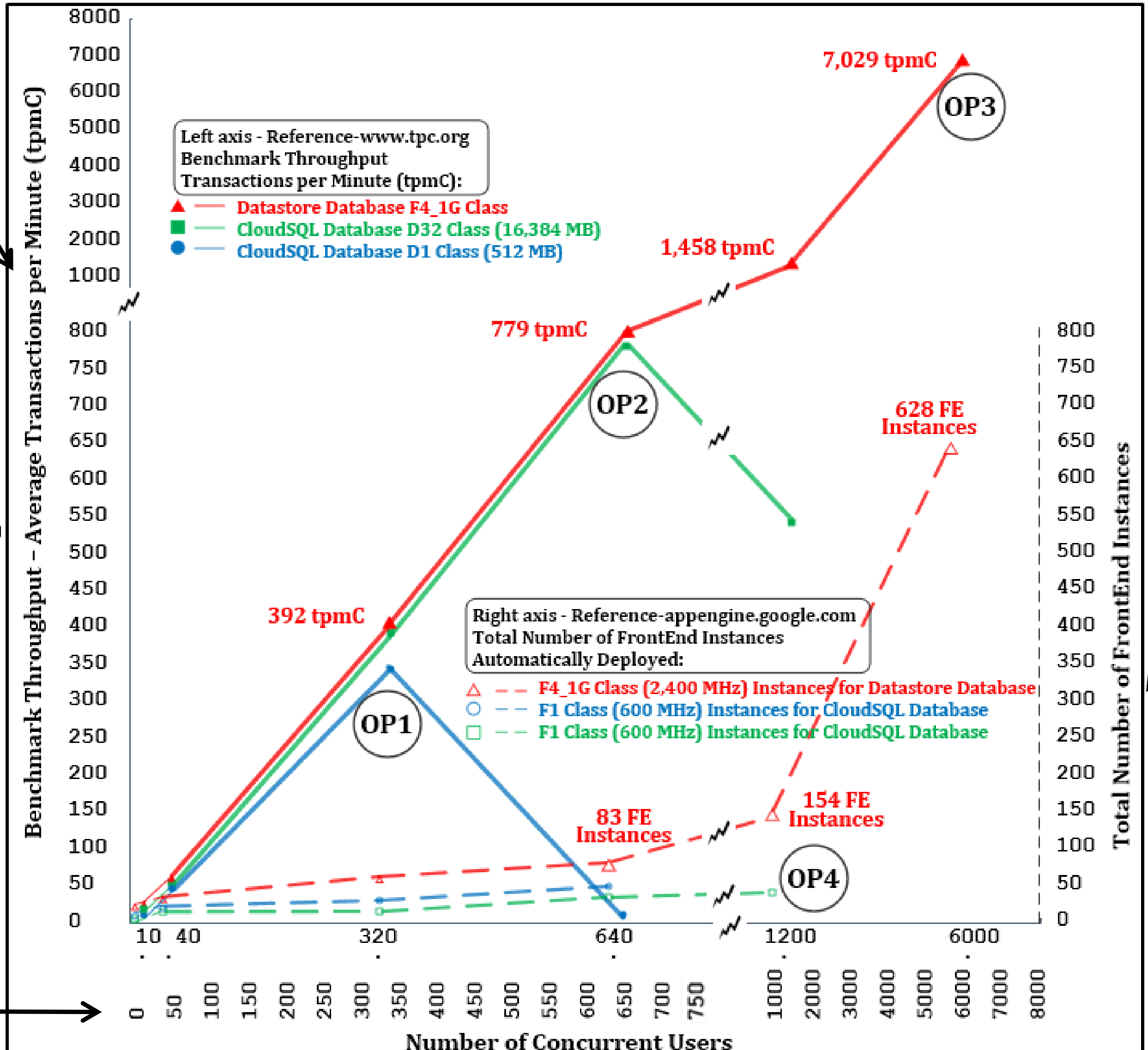
	10 users	40 users	320 users	640 users
Benchmark CloudSQL DB Instance D1 (0.5 GB Ram,$0.10/Hour) tpmC	12.5	50.4	338.1	3
CloudSQL Maximum Number of FrontEnd Instances (F1 $0.08/Hour/Instance) 128MB MEM, 600MHZ CPU	1.0	3.0	12.0	16.0

The number of Front End instances allocated by the Google Cloud with pricing

GoBench TPC-C Benchmark(s) Results

Step 4 Google Cloud Results for 15 Benchmark Experiments

Benchmark throughput in transactions per minute (tpmC)



Number of Cloud FrontEnd instances automatically allocated

The number of concurrent users, generating transactions

GoBench TPC-C Benchmark(s) Results

Step 4 Google Cloud Results for 16 Benchmark Experiments Table

Additional experiment with 640 users, CloudSQL mid-power instance D16

	10 users	40 users	320 users	640 users	1,200 users	6,000 users
CloudSQL DB Instance D1 (0.5 GB Ram,$0.10/Hour) tpmC	12.5	50.4	338.1	3		
CloudSQL DB Instance D16 (8 GB Ram,$1.54/Hour) tpmC				760.4		
CloudSQL DB Instance D32 (16 GB Ram,$3.08/Hour) tpmC	12.5	50.4	374.6	769.5	557.8	
Datastore F4_1G (2400MHZ,1024MB) tpmC	*12.0*	*49.1*	*392.3*	*778.7*	*1,458.1*	*7,028.7*
Maximum Benchmark tpmC	12.9	51.4	411.5	823.0	1,543.2	7,716.0
CloudSQL Maximum Number of FrontEnd Instances (F1 $0.08/Hour/Instance)	1.0	3.0	12.0/6.0	16.0/13.0/12.0	32.0	
Datastore Maximum Number of FrontEnd Instances (F4_1G $0.48/Hour/Instance)	*5.0*	*8.0*	*55.0*	*83.0*	*154.0*	*628.0*
Number Warehouses	1	4	32	64	120	600
CloudSQL-Database Size Gbytes ($0.24/GB/Mo)	0.2	0.5	3.3	5.7	9.9	
Datastore - Database Size Gbytes ($0.18/GB/Mo)	*2.0*	*7.5*	*47.5*	*116.8*	*228.1*	*1,368.2*

Maximum transactions per minute (tpmC) based on benchmark-required transaction keying time and think time.

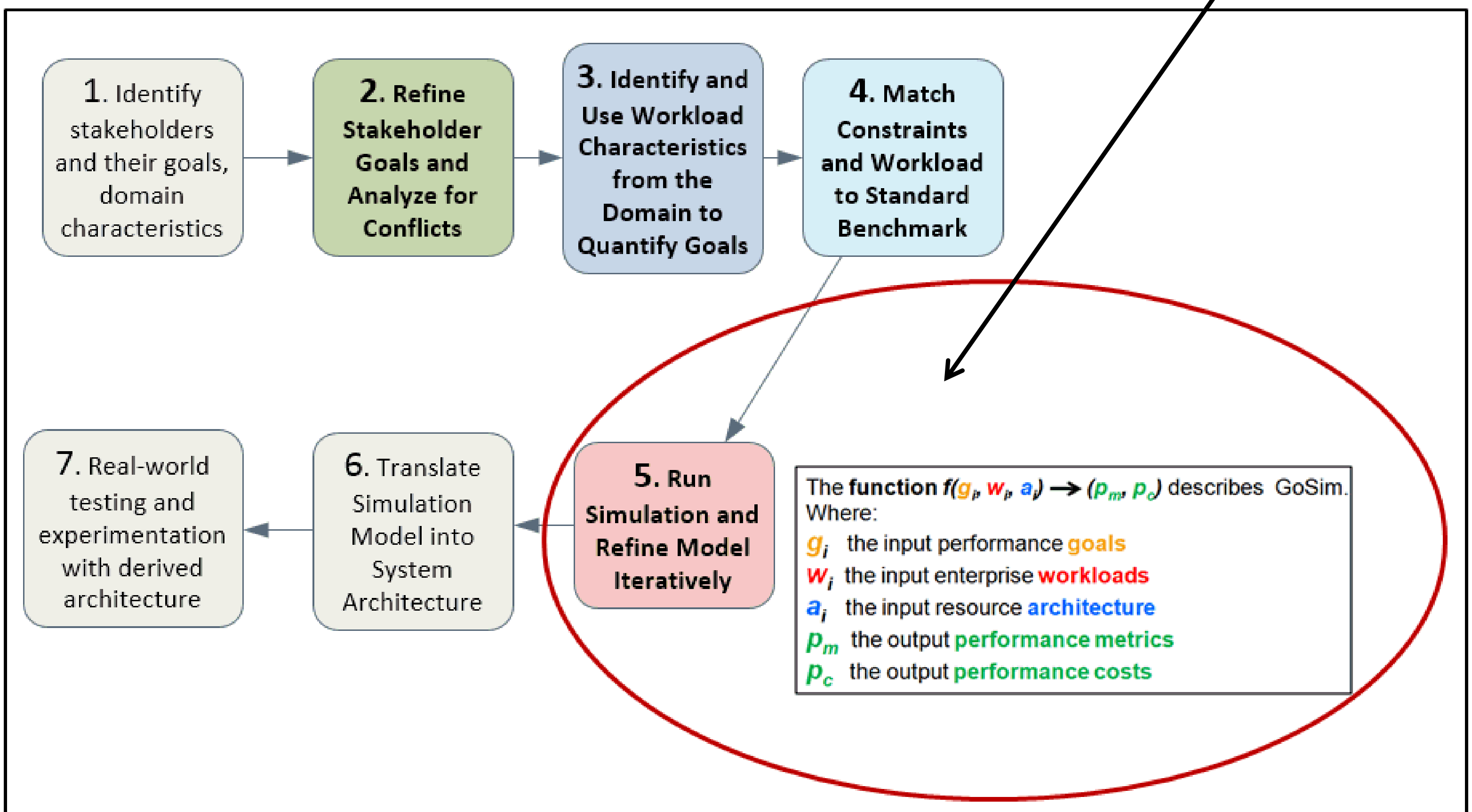
Outline

- Motivation
- Research Problem
- Related Work
- The Proposed Solution
    - GoBench
    - GoSim
- Case Studies
- Conclusion

GoSim Software Engineering Framework

Seven Software Engineering Steps - Reconfirming

Step 5 (Run Simulation Experiments GoSim Function) Highlighted



GoSim Simulation Model Three-step Process

Why Build Simulation Models?

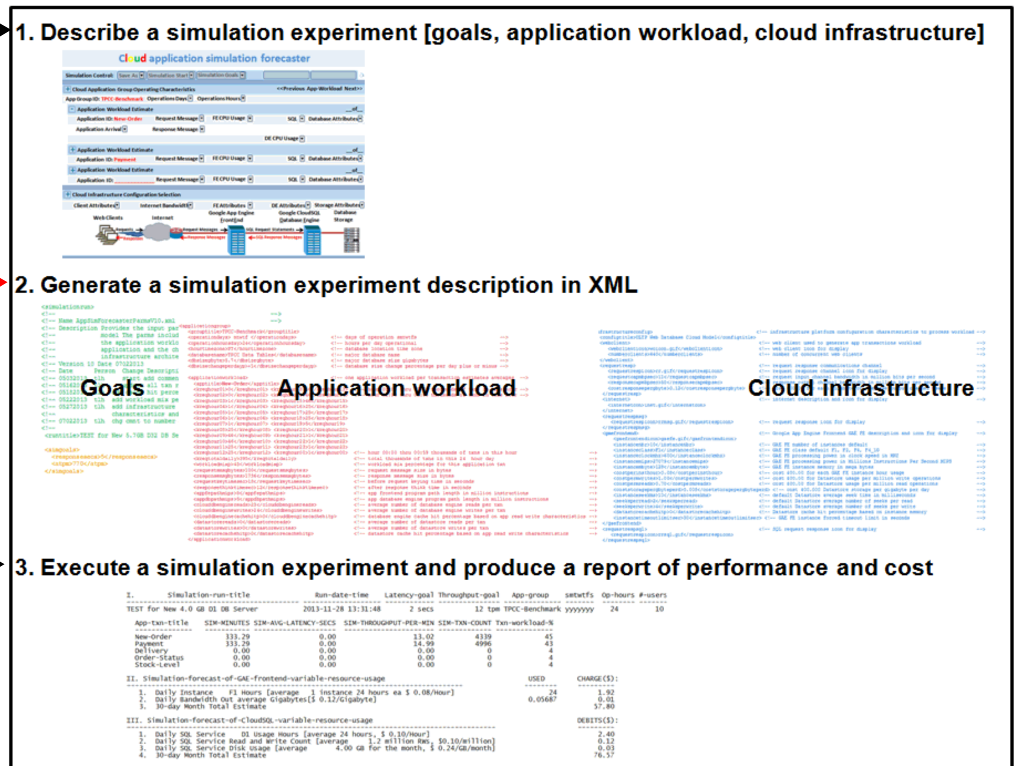
1. Understand the behavior of a complex system by describing the system, without constructing it
2. Eliminates the time and expense required to design, code and test software and build-out the hardware/software infrastructure

Simulation Three-steps (Describe Experiment, Generate, Execute):

Graphic User Interface used to design the simulation experiment; by describing: goals, application workload and the components of the infrastructure

The GUI generates the description of the simulation experiment in XML

XML is used as input to a discrete event simulator to produce a report of performance (throughput) and cost



GoSim Describe A Simulation Experiment

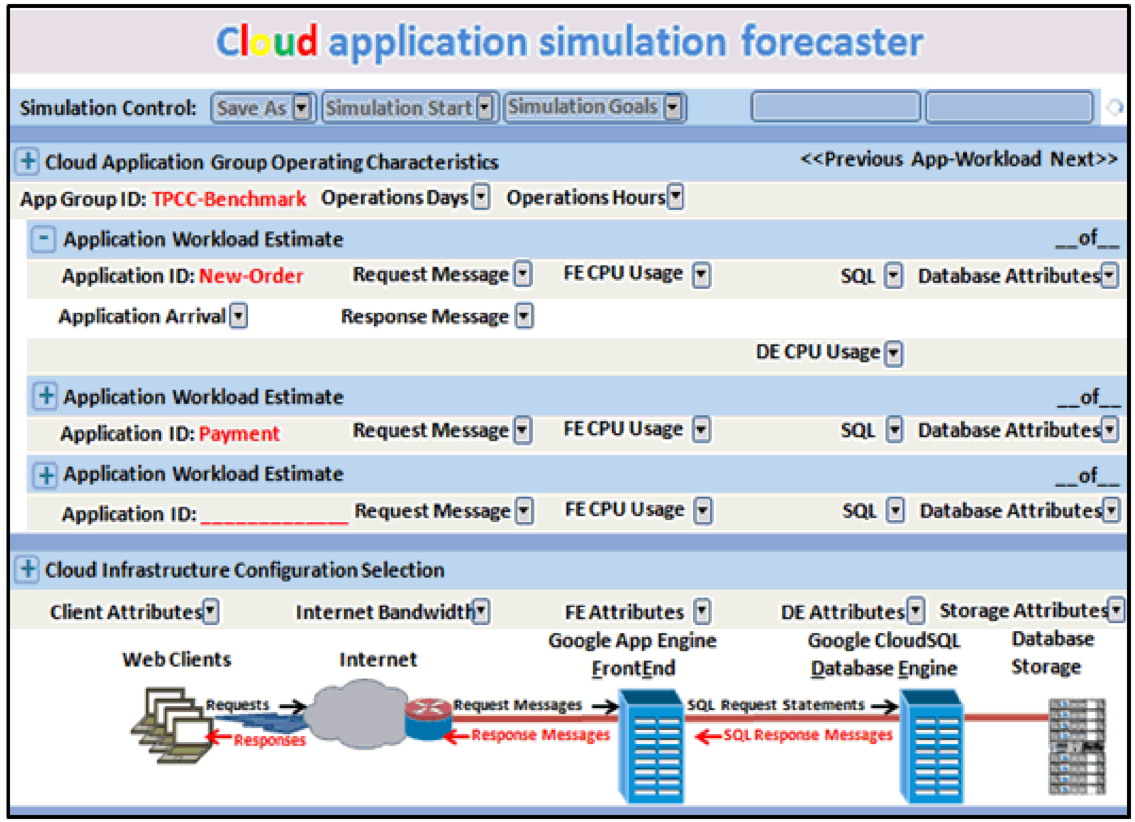
Graphical User Interface Design to Describe:

1. Performance Goals
2. Workload
3. Architecture Infrastructure

Performance goals →

Application workload →

Architecture infrastructure →



Generate XML to describe the complete simulation experiment

GoSim Describe A Simulation Experiment

XML Design to Describe:

1. Performance Goals
2. Workload
3. Architecture Infrastructure

Architecture infrastructure ↓

Performance goals →

Application workload →

Performance Goals


```
<?xml version="1.0"?>
- <simulationrun>
    <runtitle>TEST#7 for Sim Datastore</runtitle>
  - <simgoals>
      <responsesecs>3</responsesecs>
        <!-- average response time in seconds -->
      <atpm>7000</atpm>
        <!-- average application txns per minute -->
      <simtype>infinite</simtype>
        <!-- simulation type infinite or finite -->
    </simgoals>
  + <applicationgroup>
  + <infrastructureconfig>
</simulationrun>
```


Application Workload


```
- <applicationworkload>
      <!-- one application workload per transaction estimates averages -->
    <apptitle>New-Order</apptitle>
    <kreqtotaldaily>395</kreqtotaldaily>
      <!-- total thousands of txns in this 24 hour day -->
    <workloadmixp>45</workloadmixp>
      <!-- workload mix percentage for this application txn -->
    <requestsmsgbytes>100</requestsmsgbytes>
      <!-- request message size in bytes -->
    <responsemsgbytes>1736</responsemsgbytes>
      <!-- response message size in bytes -->
    <requestkeytimesec>18</requestkeytimesec>
      <!-- before request keying time in seconds -->
    <responsethinktimesec>12</responsethinktimesec>
      <!-- after response think time in seconds -->
    <appfepathmip>5</appfepathmip>
      <!-- app frontend program path length in million instructions -->
    <appdbpathmip>5</appdbpathmip>
      <!-- app database engine program path length in million instructions
    <clouddbenginereads>23</clouddbenginereads>
      <!-- average number of database engine reads per txn -->
    <clouddbenginewrites>24</clouddbenginewrites>
      <!-- average number of database engine writes per txn -->
    <clouddbenginecachehitp>0</clouddbenginecachehitp>
      <!-- database engine cache hit percentage based on app read write cl
    <datastorereads>30</datastorereads>
      <!-- average number of datastore reads per txn -->
    <datastorewrites>176</datastorewrites>
      <!-- average number of datastore writes per txn -->
    <datastorecachehitp>0</datastorecachehitp>
      <!-- datastore cache hit percentage based on app read write characteristics
. . .
```


Architecture Infrastructure


```
- <infrastructureconfig>
      <!-- infrastructure platform configuration characteristics to process workload -->
    <configtitle>OLTP Web Database Cloud Model</configtitle>
  - <webclient>
        <!-- web client used to generate app transactionw workload -->
      <webclienticon>wcicon.gif</webclienticon>
        <!-- web client icon for display -->
      <numberclients>6000</numberclients>
        <!-- number of concurrent web clients -->
    </webclient>
  - <requestresp>
        <!-- request response communications channel -->
      <requestrespicon>rr.gif</requestrespicon>
        <!-- request response channel icon for display -->
      <wanrttms>1</wanrttms>
        <!-- WAN RoundTrip Time milliseconds Ping UTD lab 1 ms, home 33 ms -->
      <requestcapmbpsec>11</requestcapmbpsec>
        <!-- request input channel bandwidth in million bits per second -->
      <responsecapmbpsec>50</responsecapmbpsec>
        <!-- response output channel bandwidth in million bits per second -->
      <costresponsepergbyte>0.12</costresponsepergbyte>
        <!-- cost $00.00 for each gigabyte of response output channel usage -->
    </requestresp>
  + <internet>
  + <requestrespmsg>
  - <gaefrontend>
        <!-- Google App Engine Frontend GAE FE description and icon for display -->
      <gaefrontendicon>gaefe.gif</gaefrontendicon>
      <instancefac>0.05</instancefac>
        <!-- GAE FE instances factor, calc nbr instances = fac * TPM -->
      <instancedpn>7</instancedpn>
        <!-- GAE FE instances datapoint number collected TPM and FrontEnds -->
      <instanceclass>F4_1G</instanceclass>
        <!-- GAE FE class default F1, F2, F4, F4_1G -->
      <instancepowermult>6.0</instancepowermult>
        <!-- GAE FE processing power and cost multiplier, with F1 as base -->
      <instancemips>27079</instancemips>
        <!-- GAE FE processing power in Millions Instructions Per Second MIPS -->
      <instancembyte>128</instancembyte>
        <!-- GAE FE instance memory in mega bytes -->
      <costperinsthour>0.08</costperinsthour>
. . .
```

GoSim Execute Simulation Forecaster

Google Cloud Project Simulation Forecaster Function Design

Use XML that describes the experiment as function input and output performance metrics and cost

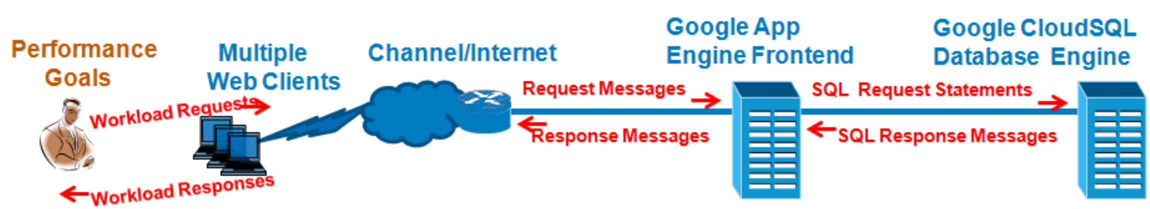
1. Encode a Simulation / Forecaster Function

The **function** $f(g_i, w_i, a_i) \rightarrow (p_m, p_c)$ describes Simulation/Forecaster. Where:

- g_i **the input stakeholder performance goals**
- w_i **the input enterprise workloads**
- a_i **the input resource architecture**
- p_m **the output performance metrics**
- p_c **the output performance costs**

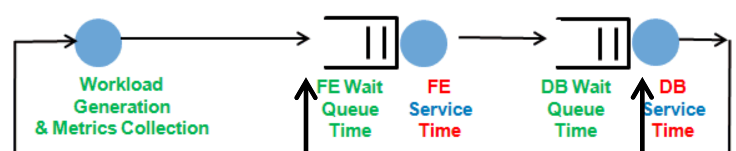
Workload requests represent resource usage and architecture infrastructure components represent capacity

2. Model the Application Workload Resource Usage and Architecture Resource Capacity



Simulator generates workload of multiple users and collects metrics

3. Construct a Simulation/Forecaster to Produce a Performance Metrics and Costs Report

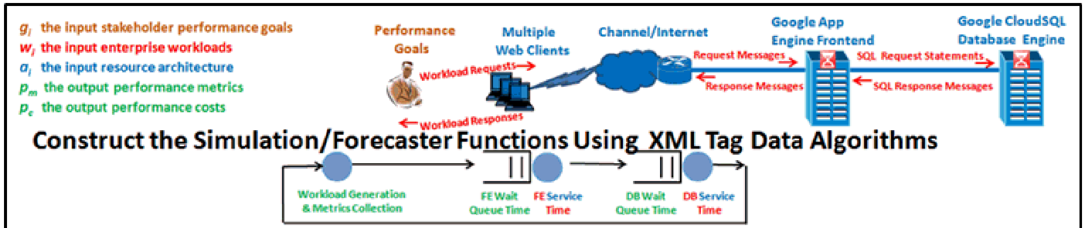


Queues are created for a finite capacity model

Service times combine usage and capacity

GoSim Execute Simulation Forecaster

Google Cloud Project Simulation Forecaster Mean Interarrival Algorithm Example



Mean Interarrival Seconds Algorithm Example – Workload Generation

New-Order App Transaction Minimum Seconds for Single User =
(<workloadmixp> * (<requestkeytimesec> + <responsethinktimesec>))
(.45 * (18 + 12)) = 13.50

Payment App Transaction Minimum Seconds for Single User =
(.43 * (3 + 12)) = 6.45

Order-Status App Transaction Minimum Seconds for Single User =
(.04 * (2 + 10)) = 0.48

Delivery App Transaction Minimum Seconds for Single User =
(.04 * (2 + 5)) = 0.28

Stock-Level App Transaction Minimum Seconds for Single User =
(.04 * (2 + 5)) = 0.28

20.99 Single User Total

Mean Interarrival Seconds = Single User Total / <numberclients>

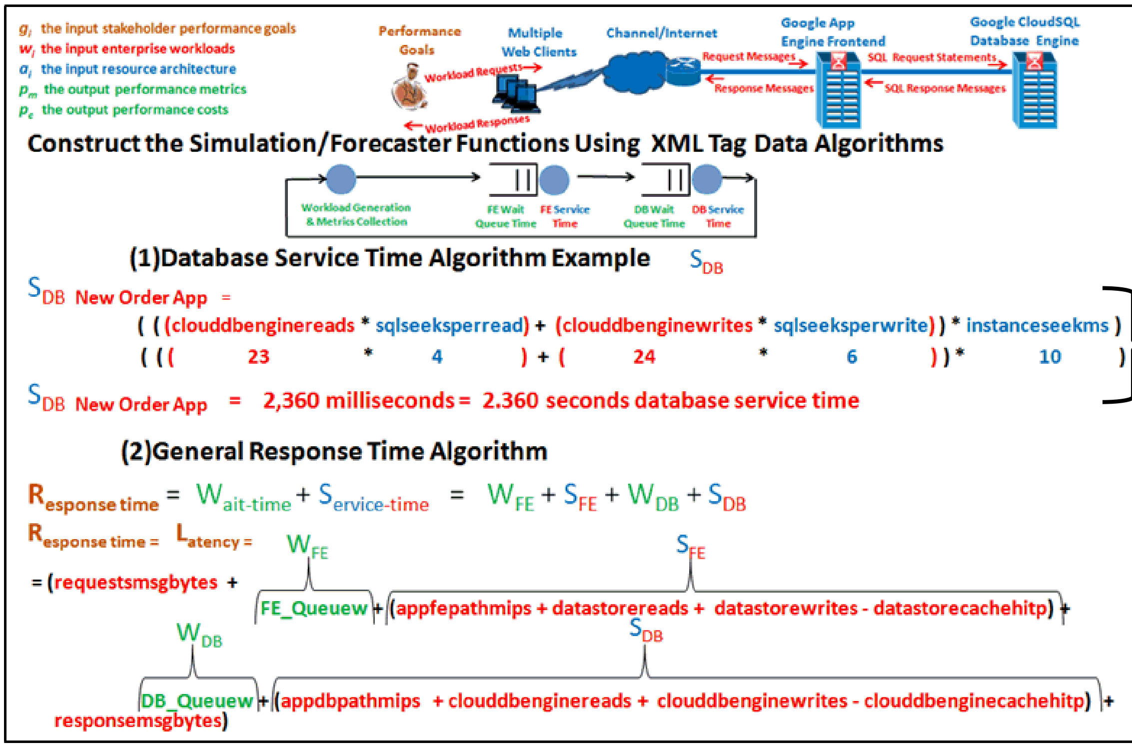
Mean Interarrival Seconds = 20.99 / 640 = 0.0328

Transaction definition for the "New Order" application workload contained in XML, includes: workload mix (45%), request keying time (18 seconds) and response think time (12 seconds)

Mean interarrival seconds for 640 users

GoSim Execute Simulation Forecaster

Google Cloud Project Simulation Forecaster Mean Database Service Time Algorithm Example



"New Order" application workload combined with cloud capacity, defined in XML, includes: mean number of cloud database read operations (23), mean number of SQL seek operations per read (4), mean number of cloud database write operations (24), mean number of SQL seek operations per write (6) and the mean seek time (10 milliseconds)

GoSim Simulation Forecaster Report

Response time goal 2 seconds

Throughput goal 300 transactions per minute

Number of concurrent users 320

```
I.       Simulation-run-title                    Run-date-time      Latency-goal Throughput-goal   App-group     smtwtfs   Op-hours #-users
-------------------------------------------- ------------------- ------------ --------------- -------------- ------- -------- -------
TEST for New 4.0 GB D1 DB Server             2014-08-02 15:30:52       2 secs         300 tpm TPCC-Benchmark yyyyyyy    24        320

  App-txn-title     SIM-MINUTES SIM-AVG-LATENCY-SECS  SIM-THROUGHPUT-PER-MIN SIM-TXN-COUNT Txn-workload-%
  -------------     ----------- --------------------  ---------------------- ------------- --------------
  New-Order              333.33                 0.00                  378.58        126194             45
  Payment                333.33                 0.00                  444.86        148284             43
  Delivery                 0.00                 0.00                    0.00             0              4
  Order-Status             0.00                 0.00                    0.00             0              4
  Stock-Level              0.00                 0.00                    0.00             0              4

II. Simulation-forecast-of-GAE-frontend-variable-resource-usage                                     USED       CHARGE($):
------------------------------------------------------------------------------------              --------   ---------
   1.  Daily Instance    F1 Hours [average  10 instance 24 hours ea $ 0.08/Hour]                      240        19.20
   2.  Daily Bandwidth Out average Gigabytes[$ 0.12/Gigabyte]                                        1.668         0.20
   3.  30-day Month Total Estimate                                                                               582.00

III. Simulation-forecast-of-CloudSQL-variable-resource-usage                                                 DEBITS($):
----------------------------------------------------------------------------------------                   ---------
   1.  Daily SQL Service   D1  Usage Hours [average 24 hours, $ 0.10/Hour]                                         2.40
   2.  Daily SQL Service Read and Write Count [average    35.2 million RWs, $0.10/million]                        3.52
   3.  Daily SQL Service Disk Usage [average       4.00 GB for the month, $ 0.24/GB/month]                        0.03
   4.  30-day Month Total Estimate                                                                              178.65
```

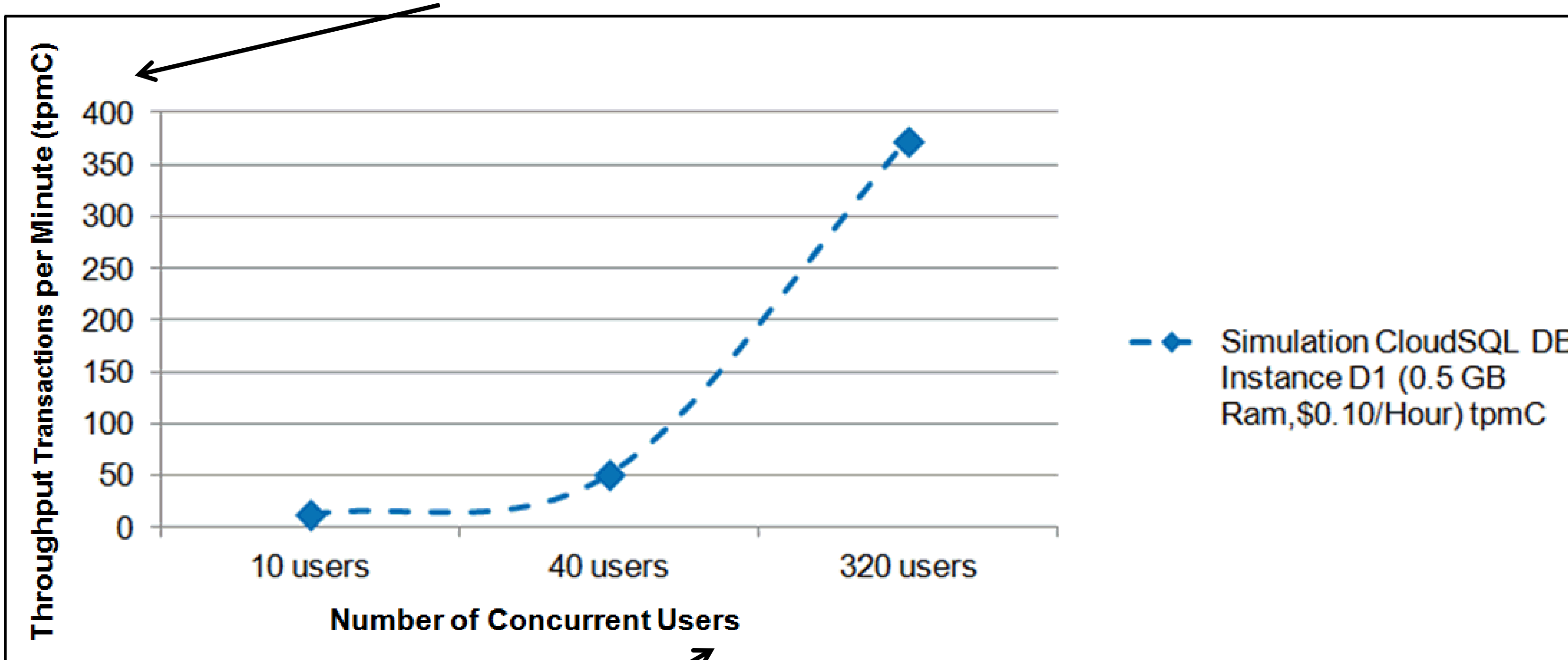
New Order application throughput 378.58 transactions per minute (tpmC)

30-day cost for GAE instances $582.00

30-day cost for Database $178.65

GoSim Throughput Simulation Results

Transactions per minute (tpmC) range 12 - 379



Three data points for number of concurrent users 10, 40, 320

Simulation results compare favorably with benchmark results

- **Benchmark throughput transactions per minute (12.5, 50.4, 338.1)**
- **Simulation throughput transactions per minute (12, 52, 379)**

Outline

- Motivation
- Research Problem
- Related Work
- The Proposed Solution
  - GoBench
  - GoSim
- Case Studies
- Conclusion

Google-Cloud-Grant Case Experiment Design

1. Build CIO Tools to help understand cloud performance and costs
2. Use standard benchmarks to test the fidelity of simulation models
3. Provide traceability from Problem to Contribution to Future Work

Project Plan Tasks:

a. Use TPC-C benchmark specification for txn performance, cost, database, workload →

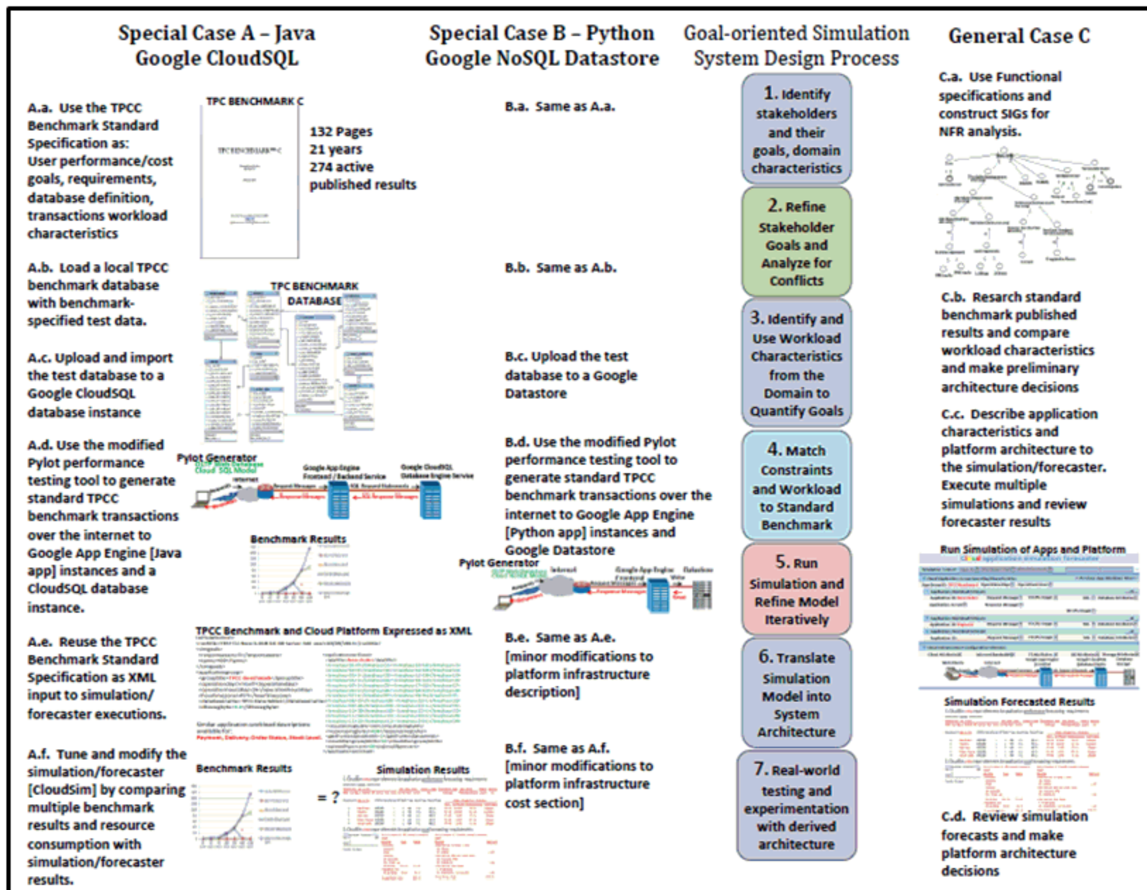
b. Generate benchmark database →

c. Upload benchmark database to cloud →

d. Use modified performance test tool to generate benchmark transactions & save results →

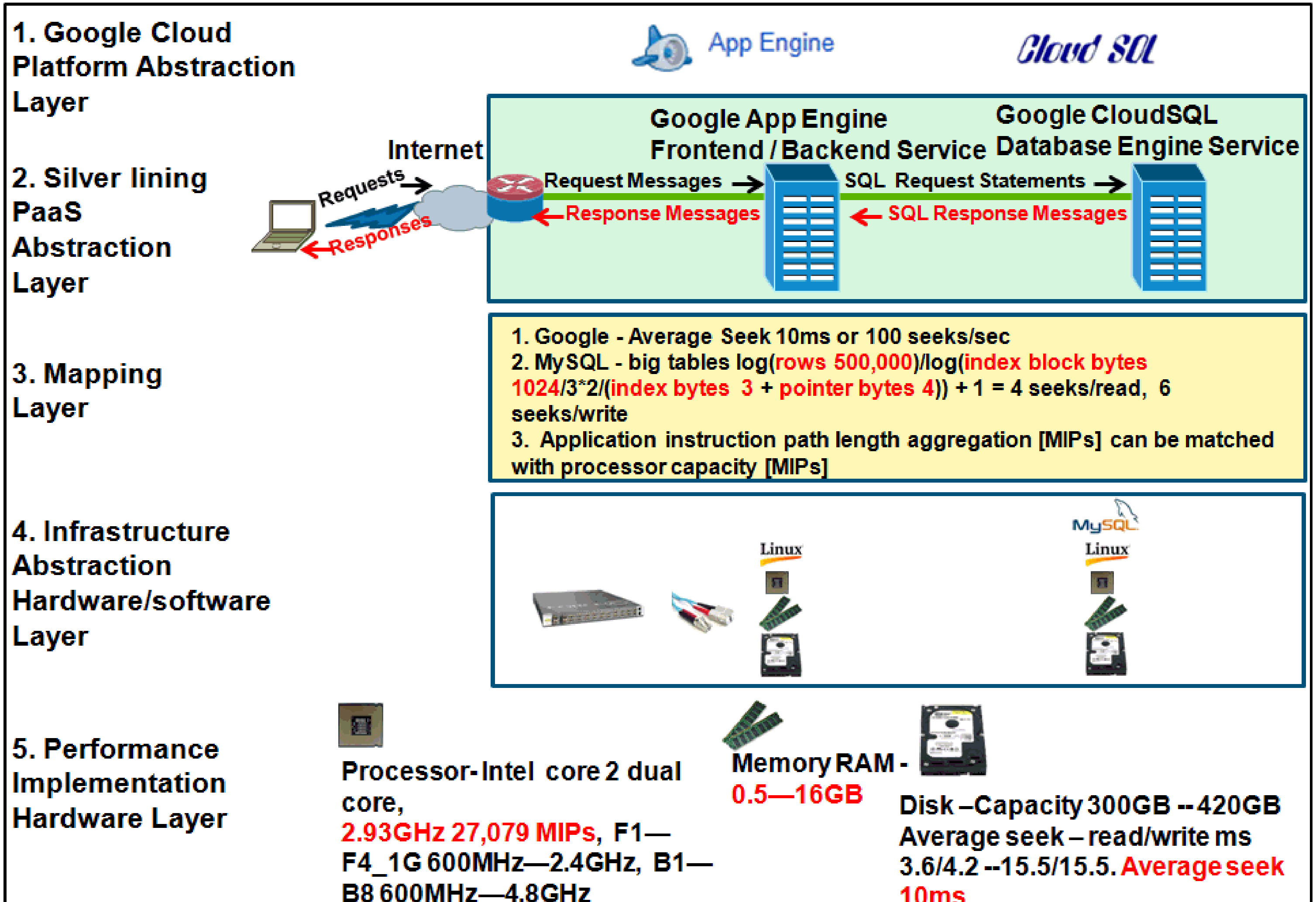
e. Restate TPC benchmark specs as XML for simulation input →

f. Run simulation / forecaster to produce performance-cost report and compare to benchmark for fidelity →



Google Cloud Infrastructure Abstraction Layers

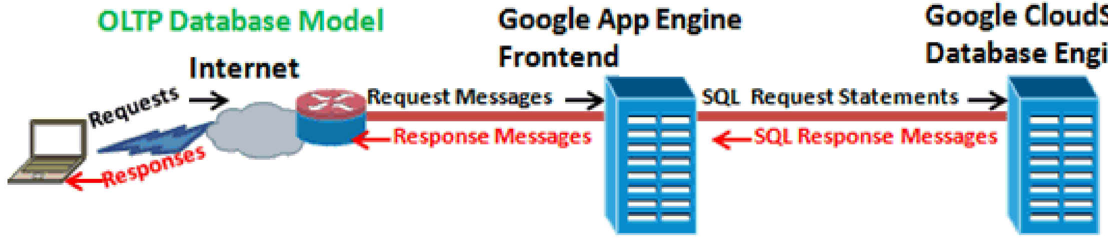
Five Layers of Discovery



Google Cloud Database Configuration Alternatives

Database CloudSQL and Datastore:

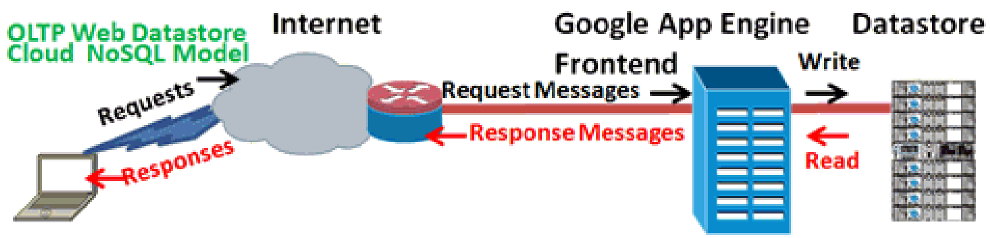
1. Google App Engine FrontEnd(s) with One CloudSQL Database Server



D1- 512MB $0.10/hour
D2- 1,024MB $0.19
D4- 2,048MB $0.38
D8- 4,096MB $0.77
D16- 8,192MB $1.54
D32- 16,384MB $3.08
(Range 1:30)

F1- 128MB 600MHz $0.08/hour
F2- 256MB 1,200MHz $0.16
F4- 512MB 2,400MHz $0.32
F4_1G 1024MB 2,400MHz $0.48
(Range 1:6)

2. Google App Engine FrontEnd(s) with Datastore Database Embedded

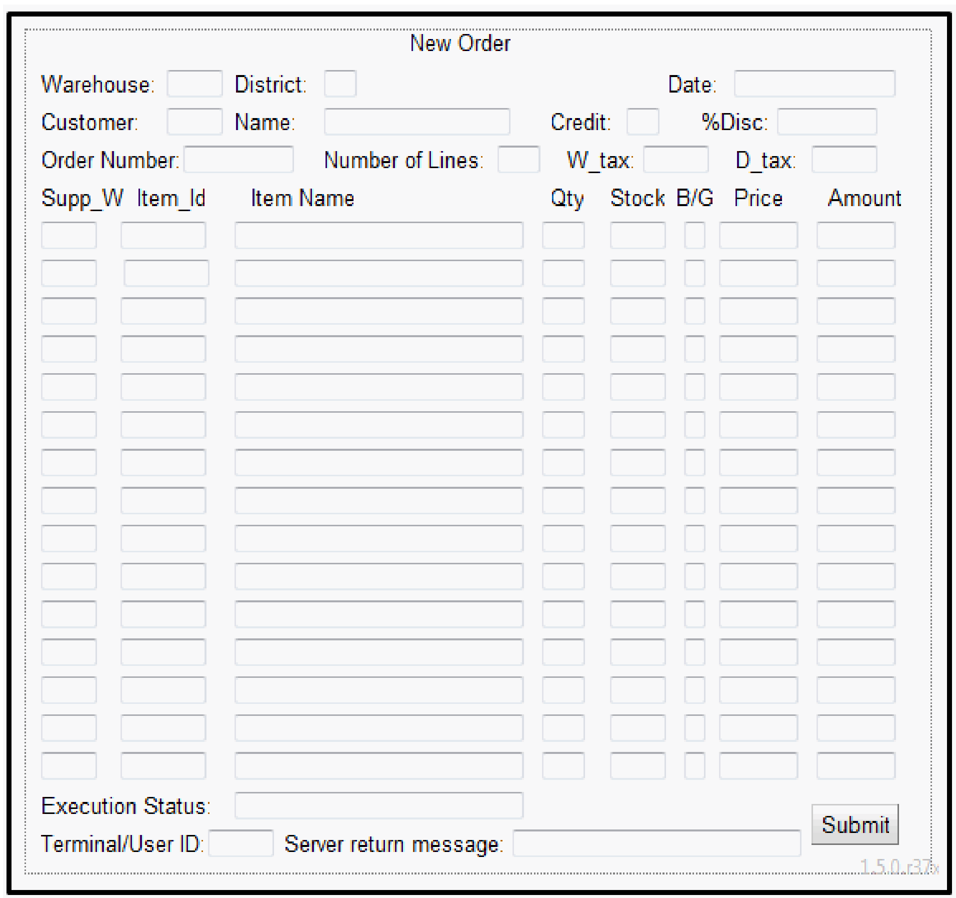


F1- 128MB 600MHz $0.08/hour
F2- 256MB 1,200MHz $0.16
F4- 512MB 2,400MHz $0.32
F4_1G 1024MB 2,400MHz $0.48
(Range 1:6)

Google Cloud TPC-C Implementation

TPC-C Benchmark Programs Re-written in Java (2 programs, 2,100 LoC) and Python (13 programs, 4,100 LoC)

Input keys randomized for benchmark generation



1. Design the New Order transaction user interface in accordance with TPC-C benchmark specifications

2. Design and implement the New Order benchmark web program in Java and Python

3. Design and implement the remaining TPC-C benchmark transactions: Payment, Order Status, Delivery, Stock Level

Google Cloud TPC-C Database Build

TPC-C Benchmark Standards Dictate Initial Database Load Characteristics

Data tables and required relationships →

```
USE tpcc32;
-- Table structure for table `customer`
--

DROP TABLE IF EXISTS `customer`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `customer` (
  `c_w_id` int(11) NOT NULL,
  `c_d_id` int(11) NOT NULL,
  `c_id` int(11) NOT NULL,
  `c_discount` decimal(4,4) DEFAULT NULL,
  `c_credit` char(2) DEFAULT NULL,
  `c_last` varchar(16) DEFAULT NULL,
  `c_first` varchar(16) DEFAULT NULL,
  `c_credit_lim` decimal(12,2) DEFAULT NULL,
  `c_balance` decimal(12,2) DEFAULT NULL,
  `c_ytd_payment` float DEFAULT NULL,
  `c_payment_cnt` int(11) DEFAULT NULL,
  `c_delivery_cnt` int(11) DEFAULT NULL,
  `c_street_1` varchar(20) DEFAULT NULL,
  `c_street_2` varchar(20) DEFAULT NULL,
  `c_city` varchar(20) DEFAULT NULL,
  `c_state` char(2) DEFAULT NULL,
  `c_zip` char(9) DEFAULT NULL,
  `c_phone` char(16) DEFAULT NULL,
  `c_since` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_
  `c_middle` char(2) DEFAULT NULL,
  `c_data` varchar(500) DEFAULT NULL,
  PRIMARY KEY (`c_w_id`,`c_d_id`,`c_id`),
  KEY `ndx_customer_name` (`c_w_id`,`c_d_id`,`c_last`,`c_first`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `customer`
--

LOCK TABLES `customer` WRITE;
/*!40000 ALTER TABLE `customer` DISABLE KEYS */;
INSERT INTO `customer` VALUES (1,1,1,0.4020,'GC','CALLYBARPRI','bglvNIwHCup
INSERT INTO `customer` VALUES (1,1,1784,0.4818,'GC','EINGPRESOUGHT','hCfXbz
INSERT INTO `customer` VALUES (1,2,553,0.4188,'GC','EINGPRESOUGHT','JIQemOL
INSERT INTO `customer` VALUES (1,2,2332,0.1829,'GC','PRIPRESATION','LhPQcQY
INSERT INTO `customer` VALUES (1,3,1114,0.3032,'GC','ESEBARESE','MQGWoutNA'
...
```

Maintain the ratio of ten users per number of warehouses in initial database

Randomized database keys and data elements →

Google Cloud Benchmark Transaction Generator

Google Cloud Project Benchmark Modified Stress-testing Tool to Generate 40 Concurrent User's Transactions

Pylot.py, open source web stress testing tool, modified to generate TPC-C benchmark transactions with random database keys, keying time and think time

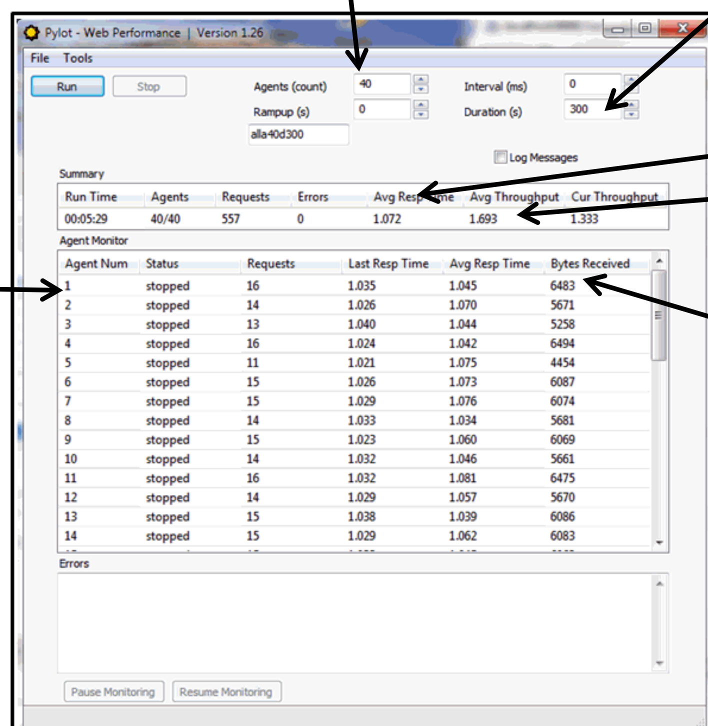
Number of concurrent users (agents) to generate transactions - 40

300 seconds benchmark duration – 300

Response time and throughput calculated and reported

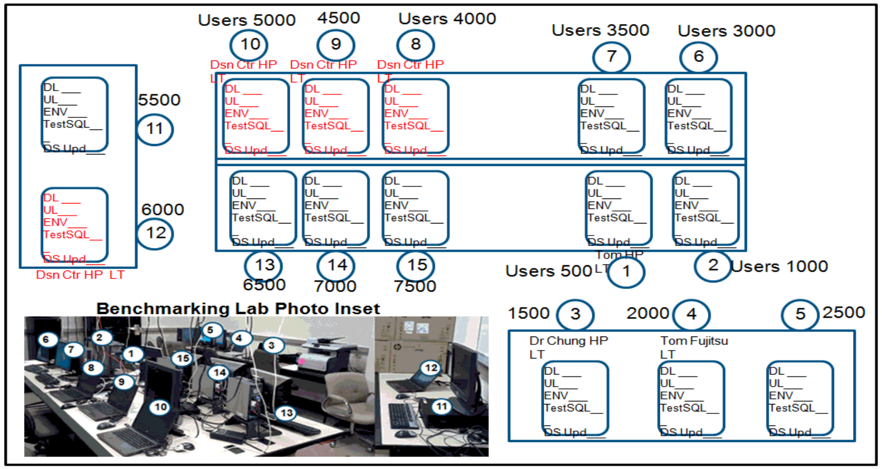
Statistics for users (agents) 1 through 40

Response message size in bytes



Google Cloud Benchmarking Infrastructure

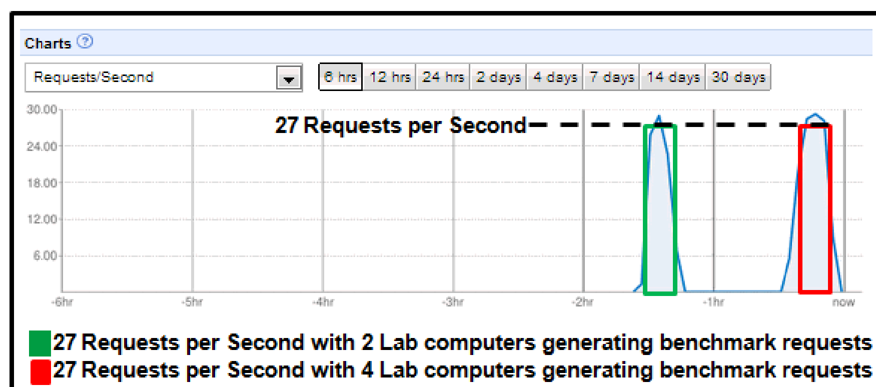
Google Cloud Project UTD Benchmark Generating Lab Map - 500 Concurrent Users through 7,500 Users



A map of 15 Lab computers generating benchmark transactions for 500 users each

Google Cloud Project Benchmark Requests per Second Strip-chart

Benchmark experiments with varying number of computers (2 versus 4 shown in the Google-provided strip-chart) generating the same total transaction volume. Test the lab sensitivity to generating environment changes. 2 computers versus 4 computers demonstrated no sensitivity (27 requests per second)



Google Cloud Benchmark Metrics

Google Cloud Project Benchmark Record of Experiments Example

10/06/2013 Test Name	Users (count)	Duration (seconds)	WHSE Start	# Users Accum	Duration Minutes	Duration Hours	Actual Start pm	# GAE FE Instances	# Requests Processed	Bandwidth Download Mbps	Bandwidth Upload Mbps	Comments
HM1	500	5100	0	500	85	1.42	3:22	255	92,660	56	20	Start of bench
HM2	500	4980	50	1,000	83	1.38	3:24	322	89,935	61	22	
HM3	500	4860	100	1,500	81	1.35	3:26	363	87,215	63	22	
HM4	500	4740	150	2,000	79	1.32	3:28	373	84,464	56	22	
HM5	500	4620	200	2,500	77	1.28	3:30	445	81,629	65	22	
HM6	500	4500	250	3,000	75	1.25	3:32	464	79,392	60	22	
HM7	500	4380	300	3,500	73	1.22	3:34	487	76,369	59	21	
HM8	500	4260	350	4,000	71	1.18	3:36	512	73,614	60	23	
HM9	500	4140	400	4,500	69	1.15	3:38	559	70,932	60	22	
HM10	500	4020	450	5,000	67	1.12	3:40	577	68,282	59	21	
HM11	500	3900	500	5,500	65	1.08	3:42	605	65,940	26	18	
HM12	500	3780	550	6,000	63	1.05	3:44	628	63,348	55	20	Level instances

6,000 concurrent users level benchmark

Lab computer name

Accumulated number of users

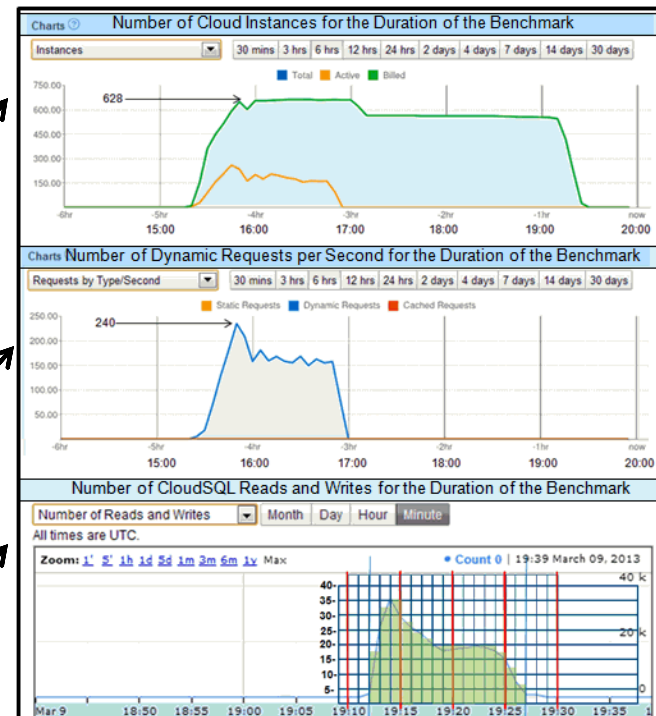
Number of Front End instances assigned by Google

Benchmark Data Collection Resource Usage Time Strip-charts Provided by Google

Cloud FE instances 628

240 requests per second

CloudSQL number of reads and writes

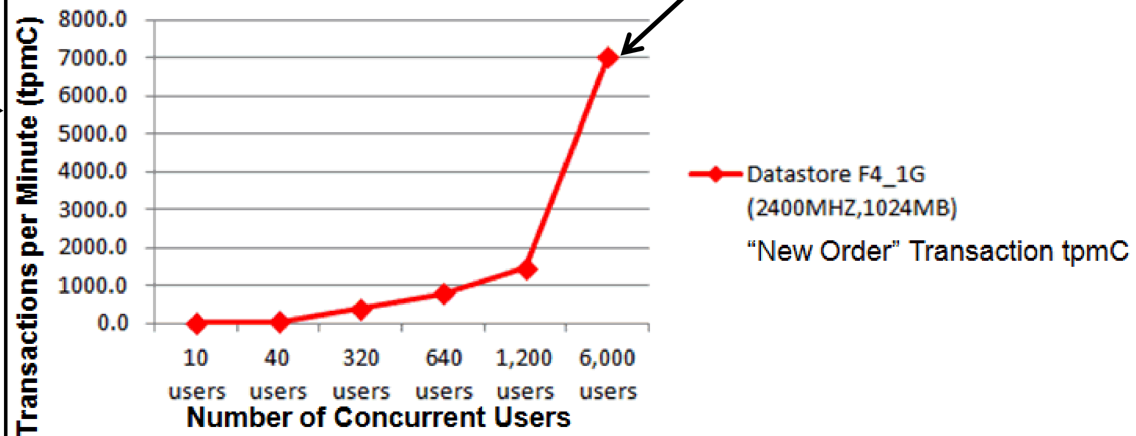


Google Cloud Benchmark Results

Google Cloud Results for Embedded Datastore Database

The performance throughput using Datastore as a database (7,028.7 transactions per minute with 6,000 concurrent users)

Benchmark throughput in transactions per minute (tpmC) for a variable number of concurrent users

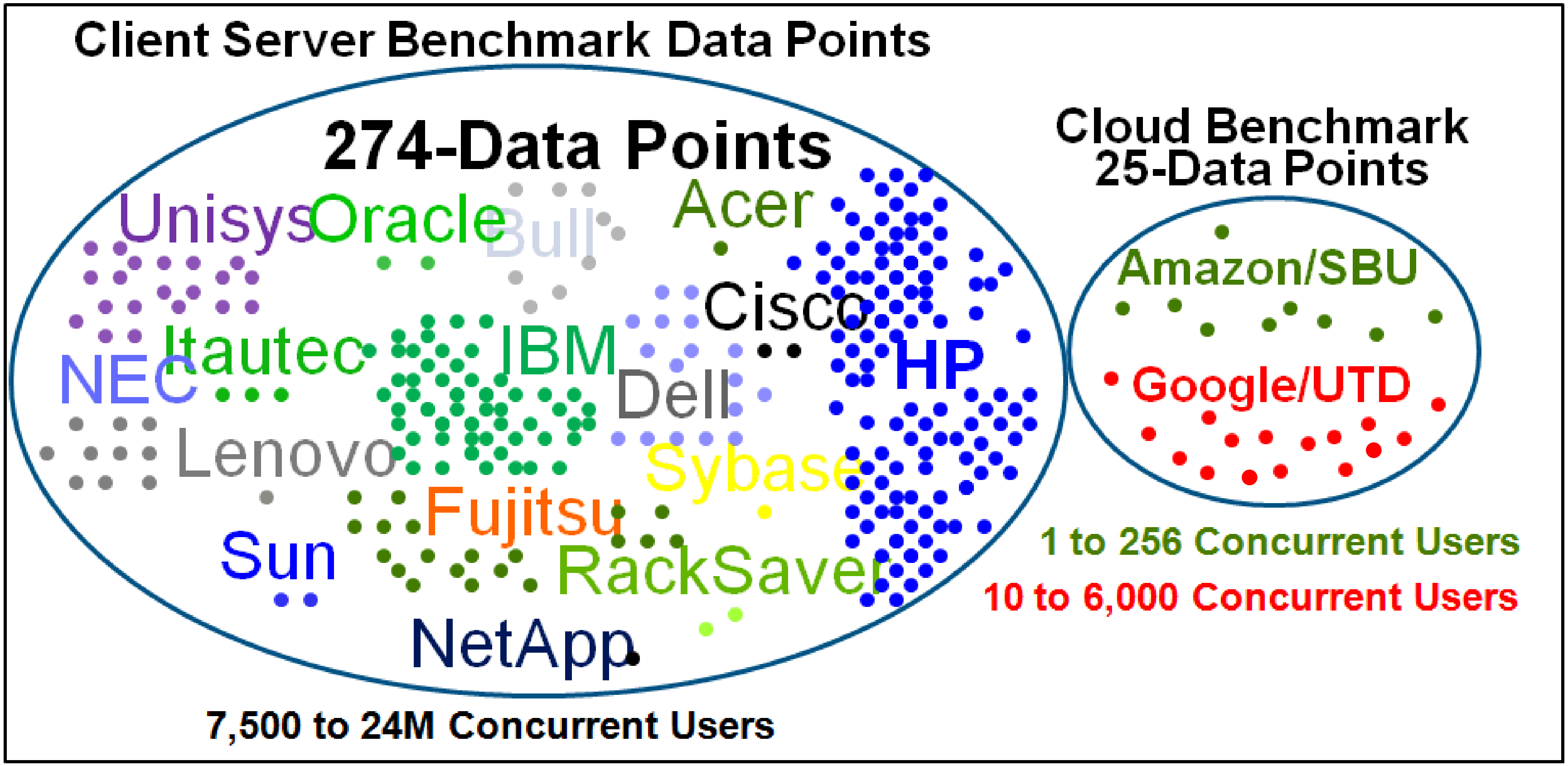


	10 users	40 users	320 users	640 users	1,200 users	6,000 users
Datastore F4_1G (2400MHZ,1024MB) tpmC	*12.0*	*49.1*	*392.3*	*778.7*	*1,458.1*	*7,028.7*
Datastore Maximum Number of FrontEnd Instances (F4_1G $0.48/Hour/Instance) 1024 MB MEM, 2400MHZ CPU	*5.0*	*8.0*	*55.0*	*83.0*	*154.0*	*628.0*

The number of Front End instances allocated by the Google Cloud with pricing

Google Cloud Benchmark Versus Client Server Results

Maximum Number of Concurrent Users in Cloud Increased to 6,000



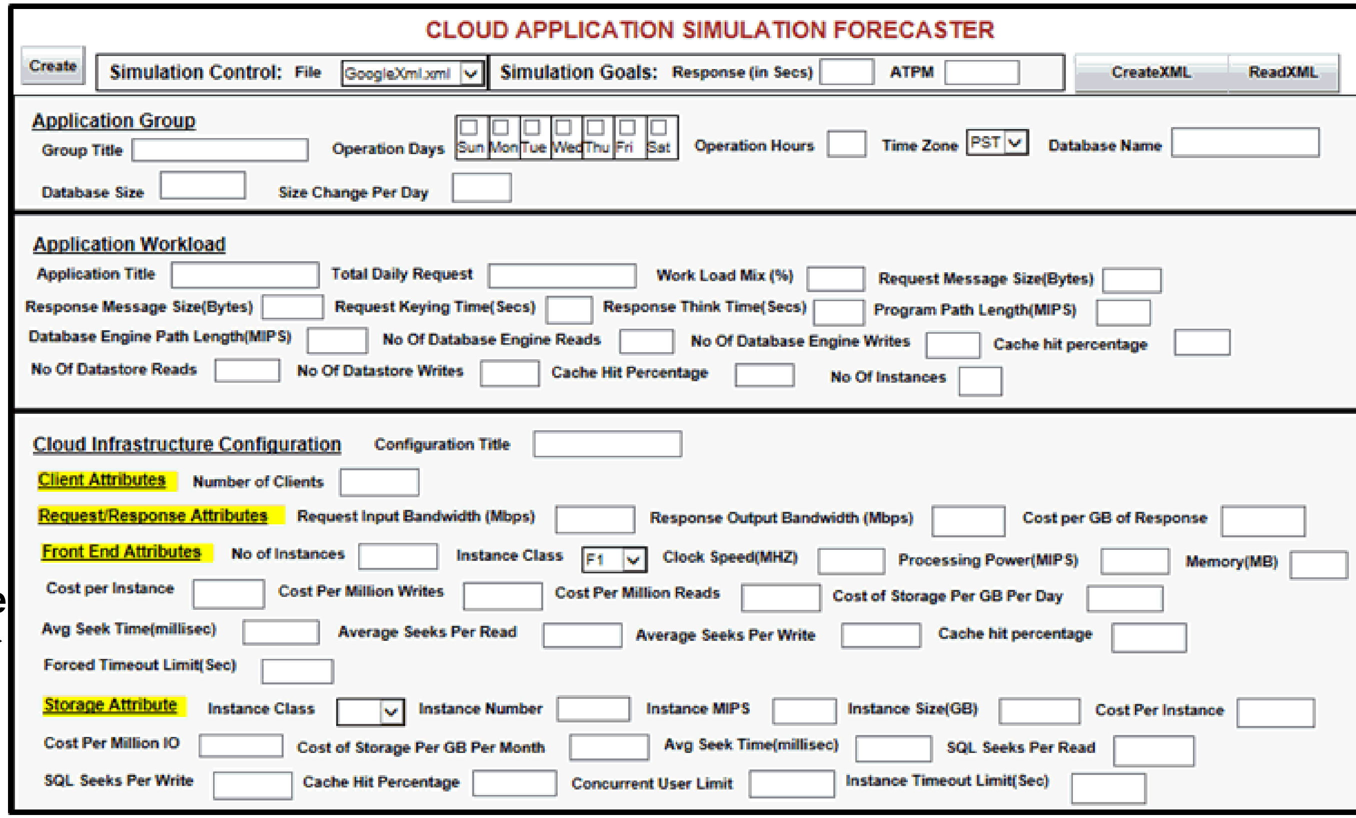
Google Cloud Simulation Describe a Simulation Experiment

Google App Script Implementation of Cloud Application Simulation Forecaster GUI

Performance goals →

Application workload →

Architecture infrastructure →



Generate XML to describe the complete simulation experiment

Google Cloud Simulation Describe a Simulation Experiment

Excerpt of a Generated XML Description of a Google Cloud Architecture Infrastructure with Component Costs Highlighted


```
. . .
<infrastructureconfig>
 <webclienticon>wcicon.gif</webclienticon>
  <numberclients>320</numberclients>
 <requestrespicon>rr.gif</requestrespicon>
  <requestcapmbpsec>11</requestcapmbpsec>
  <responsecapmbpsec>50</responsecapmbpsec>
  <costresponsepergbyte>0.12</costresponsepergbyte>
 <internetcon>inet.gif</internetcon>
 <requestrespicon>rr.gif</requestrespicon>
 <gaefrontendicon>gaefe.gif</gaefrontendicon>
  <instanceclass>F1</instanceclass>
  <instanceclockmhz>600</instanceclockmhz>
  <instancembyte>128</instancembyte>
  <costperinsthour>0.08</costperinsthour>
 <requestrespicon>rr.gif</requestrespicon>
 <clouddbengineicon>dbeng.gif</clouddbengineicon>
  <instanceclass>D1</instanceclass>
  <instancegbyte>0.5</instancegbyte>
  <costperinsthour>0.10</costperinsthour>
  <costpermio>0.10</costpermio>
 <requestrespicon>rr.gif</requestrespicon>
 <dbstorageicon>storage.gif</dbstorageicon>
  <coststoragepergbyteperm>0.24</coststoragepergbyteperm>

</infrastructureconfig>

</simulationrun>
```


The cost for bandwidth out per Giga byte [costresponsepergbyte]

The cost for a F1 GAE Frontend instance hour [costperinsthour]

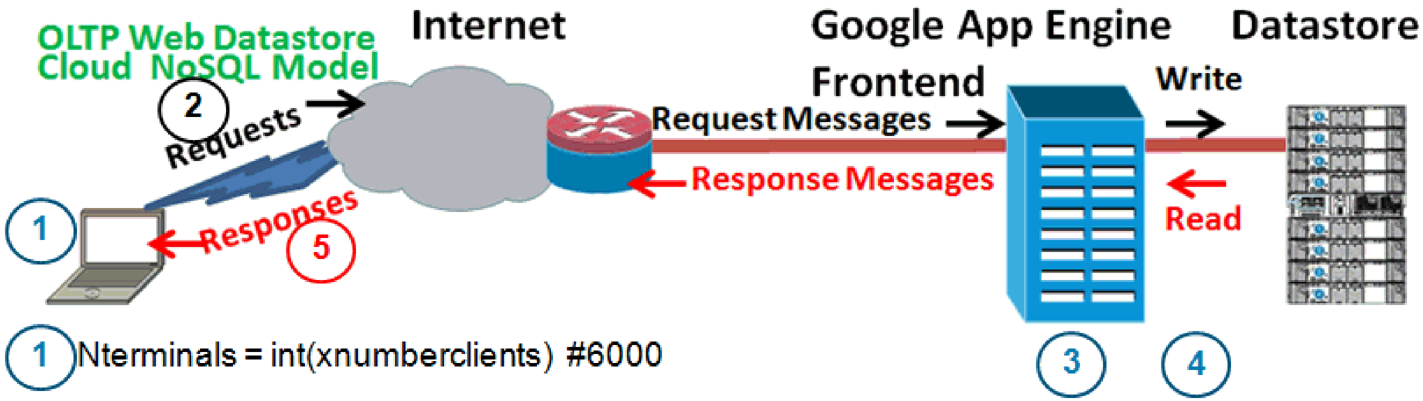
The cost for CloudSQL D1 database instance hour [costperinsthour]

The cost for CloudSQL million I/Os [costpermio]

The cost for CloudSQL storage per Giga byte per month [coststoragepergbyteperm]

Google Cloud Simulation Key Simulation Model Variables

SimPy DES Framework Simulation Model Key XML Variables



1 Nterminals = int(xnumberclients) #6000

2 keytime = 18.0 #seconds
yield hold,self,keytime

3 MeanGAETime = 1.0 #second service time
GAEtime=ran.expovariate(1.0/MeanGAETime)
yield hold,self,GAEtime

4 MeanDATASTORETime = 2.06 #seconds service time
# (<datastorereads>30+<datastorewrites>176) *10ms average seek time
DATASTOREtime=ran.expovariate(1.0/MeanDATASTORETime)
yield hold,self,DATASTOREtime

5 thinktime = 12.0 #seconds
yield hold,self,thinktime

Google Cloud Simulation Results Report

SimPy Model Report for 10 Users and 6,000 Users

```
I.        Simulation-run-title                   Run-date-time    Latency-goal Throughput-goal   App-group    smtwtfs  Op-hours #-users
-------------------------------------------  ------------------- ------------ --------------- ------------- -------- -------- -------
TEST for sim                                 2013-12-31 17:36:42       2 secs         Max tpm TPCC-Benchmark yyyyyyy     2          10

  App-txn-title     SIM-MINUTES SIM-AVG-LATENCY-SECS  SIM-THROUGHPUT-PER-MIN SIM-TXN-COUNT Txn-workload-%
  -------------     ----------- --------------------  ---------------------- ------------- -------------
  New-Order              333.32                 0.00                   11.82          3939            45
  Payment                333.32                 0.00                   13.89          4628            43
  Delivery                 0.00                 0.00                    0.00             0             4
  Order-Status             0.00                 0.00                    0.00             0             4
  Stock-Level              0.00                 0.00                    0.00             0             4

II. Simulation-forecast-of-GAE-frontend-variable-resource-usage                                      USED        CHARGE($):
-------------------------------------------------------------------------------------------------- --------    ---------
1.  Daily F4_1G 2    instances [F1 billing @    2 (F4_1G) *  6 (power) *  2 (hours/day) ea $ 0.08/Hour]      24         1.92
2.  Daily Bandwidth Out average Gigabytes[$ 0.12/Gigabyte]                                            0.004         0.00
3.  30-day Month Total Estimate                                                                                   57.62

III. Simulation-forecast-of-Datastore-variable-resource-usage                                        USED        CHARGE($):
-------------------------------------------------------------------------------------------------- --------    ---------
1.  Daily Datastore Writes $  0.9 per million operations]                                              0.31         0.28
2.  Daily Datastore Reads  $  0.6 per million operations]                                              0.06         0.04
3.  Daily Datastore Storage $0.006 per GB per day]                                                  1390.00         8.34
4.  30-day Month Total Estimate                                                                                  259.85


I.        Simulation-run-title                   Run-date-time    Latency-goal Throughput-goal   App-group    smtwtfs  Op-hours #-users
-------------------------------------------  ------------------- ------------ --------------- ------------- -------- -------- -------
TEST for sim                                 2013-12-31 17:29:52       2 secs         Max tpm TPCC-Benchmark yyyyyyy     2        6000

  App-txn-title     SIM-MINUTES SIM-AVG-LATENCY-SECS  SIM-THROUGHPUT-PER-MIN SIM-TXN-COUNT Txn-workload-%
  -------------     ----------- --------------------  ---------------------- ------------- -------------
  New-Order               52.74                 0.00                 7033.44        370925            45
  Payment                 52.74                 0.00                 8326.57        435074            43
  Delivery                 0.00                 0.00                    0.00             0             4
  Order-Status             0.00                 0.00                    0.00             0             4
  Stock-Level              0.00                 0.00                    0.00             0             4

II. Simulation-forecast-of-GAE-frontend-variable-resource-usage                                      USED        CHARGE($):
-------------------------------------------------------------------------------------------------- --------    ---------
1.  Daily F4_1G 769  instances [F1 billing @  769 (F4_1G) *  6 (power) *  2 (hours/day) ea $ 0.08/Hour]    9228       738.24
2.  Daily Bandwidth Out average Gigabytes[$ 0.12/Gigabyte]                                            2.591         0.31
3.  30-day Month Total Estimate                                                                               22156.53

III. Simulation-forecast-of-Datastore-variable-resource-usage                                        USED        CHARGE($):
-------------------------------------------------------------------------------------------------- --------    ---------
1.  Daily Datastore Writes $  0.9 per million operations]                                            187.51       168.76
2.  Daily Datastore Reads  $  0.6 per million operations]                                             38.31        22.99
3.  Daily Datastore Storage $0.006 per GB per day]                                                  1390.00         8.34
4.  30-day Month Total Estimate                                                                                 6002.67
```

Number of concurrent Users 6,000

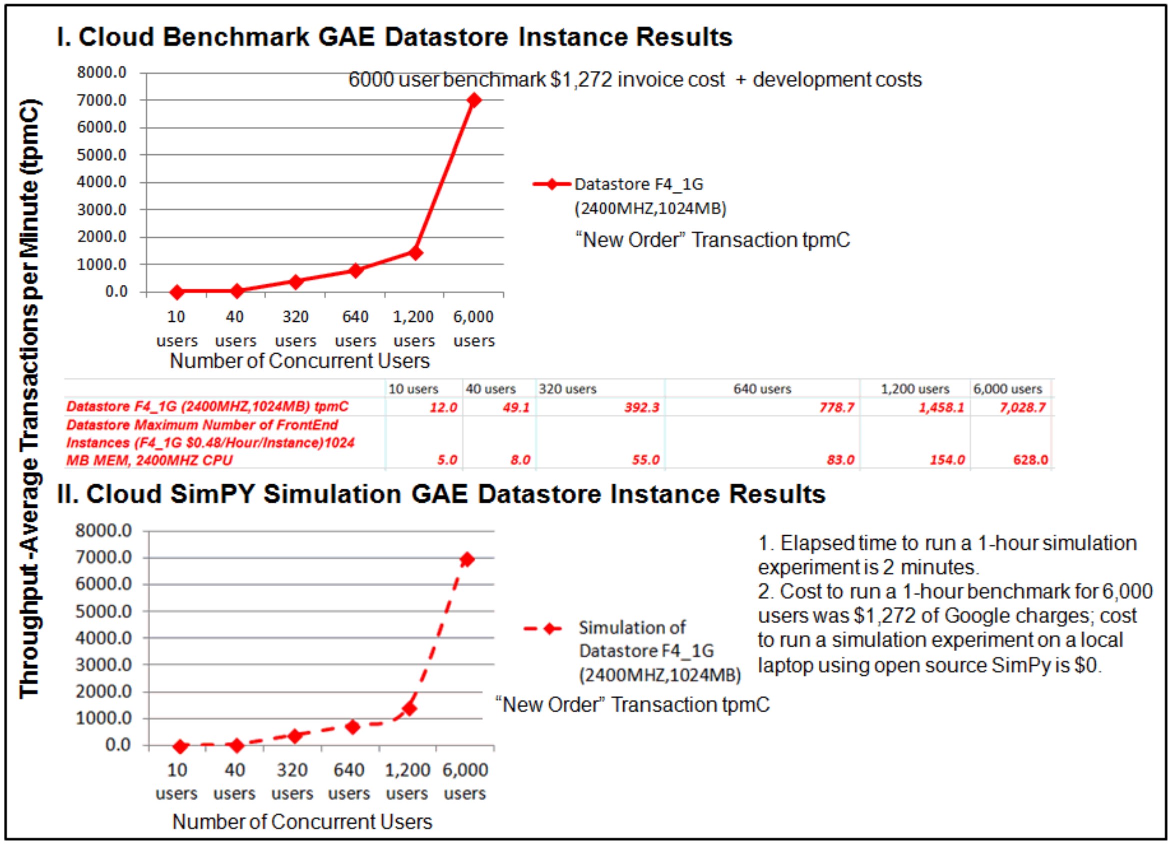
New Order application throughput 7,033.44 transactions per minute (tpmC)

30-day cost for GAE instances $22,156.53

30-day cost for Datastore database $6,002.67

Google Cloud Benchmark(s) and Simulation(s) Fidelity

Identical Number of Concurrent Users (10, 40, 320, 640, 1200, 6000) for Benchmark and Simulation



	10 users	40 users	320 users	640 users	1,200 users	6,000 users
Datastore F4_1G (2400MHZ,1024MB) tpmC	12.0	49.1	392.3	778.7	1,458.1	7,028.7
Datastore Maximum Number of FrontEnd Instances (F4_1G $0.48/Hour/Instance)1024 MB MEM, 2400MHZ CPU	5.0	8.0	55.0	83.0	154.0	628.0

Simulation results compare favorably with benchmark results

Benchmark throughput transactions per minute (12.0, 49.1, 392.3, 778.7, 1458.1, 7028.7)

Simulation throughput transactions per minute (12, 47, 379, 756, 1416, 7033)

Google Cloud Case Experiment Summary

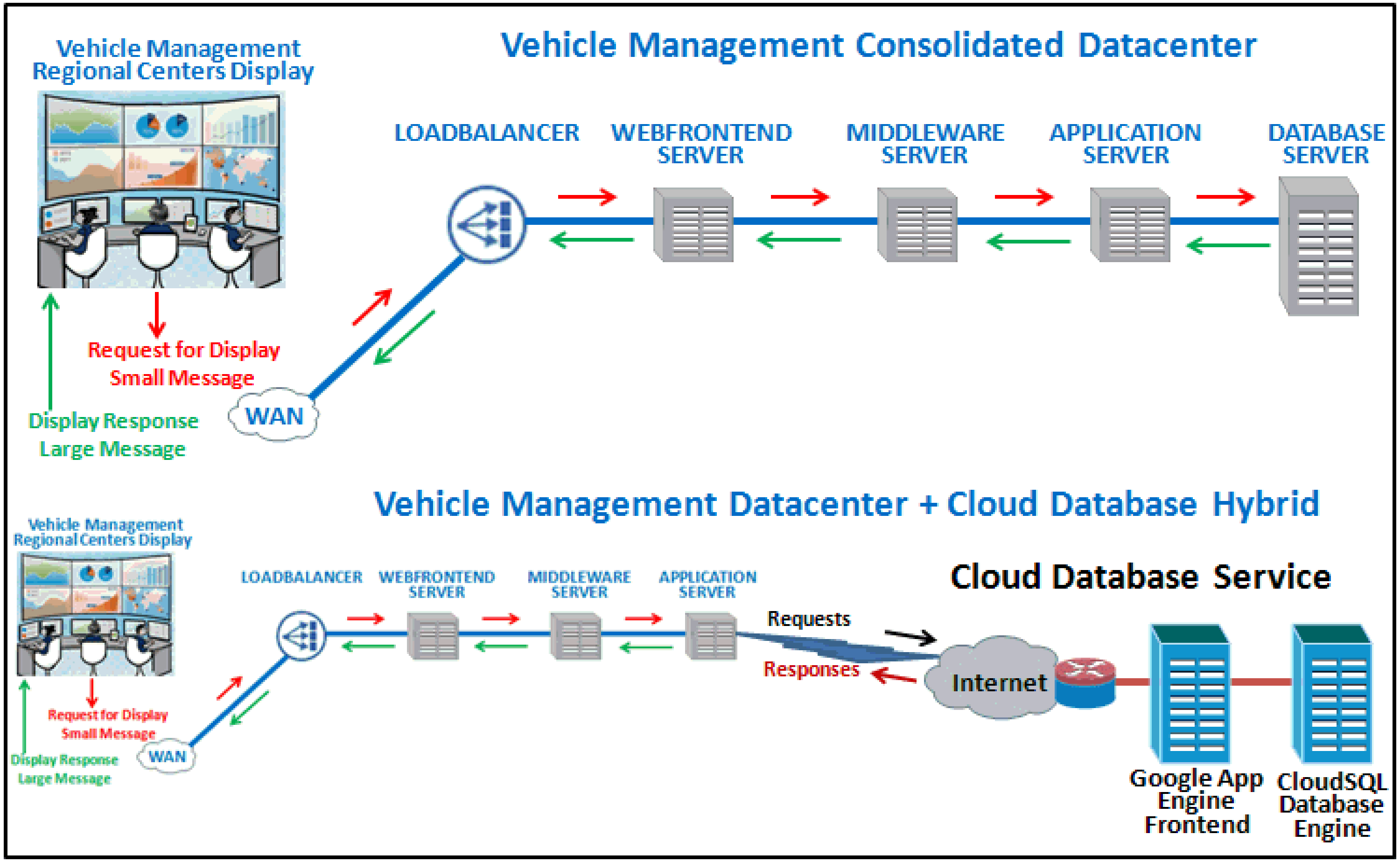
Milestone Event	Dates	Metrics/Information
Create a statistical model of the TPC-C benchmark databases and transaction workloads	10/2012	14 databases, 5 transactions
Organize Silverlining research team and deploy GAE tutorial programs locally and remotely	11/2012	21 UTD software engineering students
Design , code [Java] and test the complete TPC-C benchmark for GAE and CloudSQL	12/2012-1/2013	2 programs, 2.1 Kloc, 3 person months
Execute TPC-C benchmarks for GAE and CloudSQL	2/2013-5/2013	29 benchmark runs and analysis
Design , code [Python] and test the complete TPC-C benchmark and DMS for GAE Datastore NoSQL	10/2012-6/2013	13 programs, 4.1 Kloc, 9 person months
Begin modification of three open source discrete event simulators for GAE Simulation/forecaster	5/2013	CloudSim, Omnet++, SimPy
Design , code [Google App Script/XML] and test the graphical user interface XML generator	6/2013-7/2013	2 programs, 1.1 Kloc, 3 person months
Generate and import a TPC-C benchmark 750-warehouse-database to the GAE Datastore NoSQL	9/2013	1.3 terabytes
Execute TPC-C benchmarks for GAE Datastore NoSQL	7/2013-10/2013	11 benchmark runs and analysis

GAE Research UTD Project Milestone Table

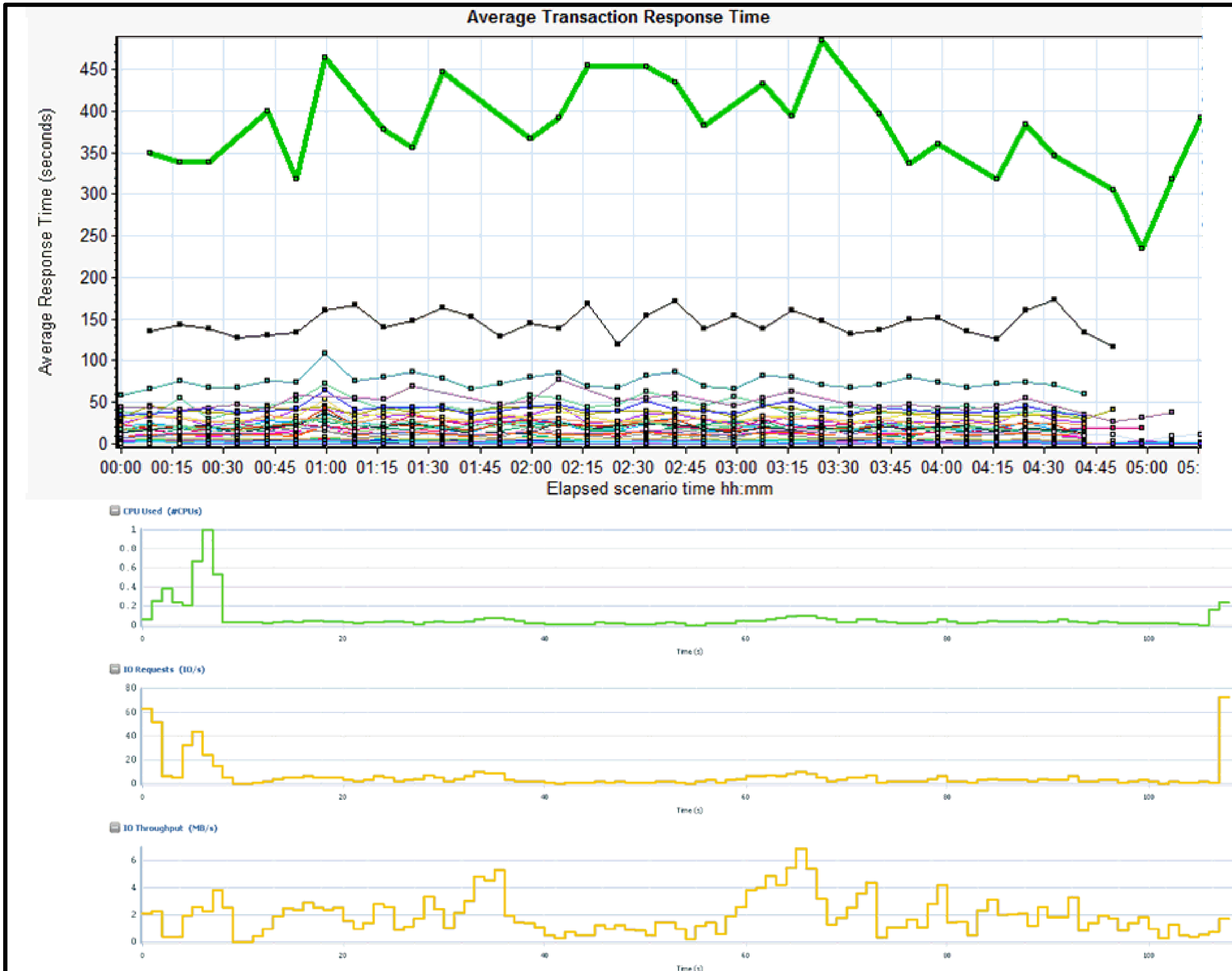
1. **Software Engineering tools (GoBench GoSim) were constructed to help CIOs understand cloud performance and costs.**
2. **Standard benchmarks (TPC-C) were re-coded for Google Cloud App Engine and used to test the fidelity of simulation models.**
3. **UTD graduate students had no difficulty re-coding the benchmarks (in Java and Python) or executing in the cloud.**
4. **Use of cloud pay-as-you-go resources proved to eliminate the need to build out an infrastructure – benchmark experiments for usage cost ranged ($0 - $1,300) per 1-hour benchmark execution [simulation experiment costs running open source SimPy on a personal laptop $0].**

Vehicle Management System (VMS)

1. Analyze VMS operations data and validate architecture for future
2. Build simulation model experiments to demonstrate feasibility of alternative infrastructure designs under consideration



VMS Operation Metrics Collection



VMS Simulations

Local Datacenter Simulation

```
MaxCompletions                 =     800000
MaxrunTime minutes             =     333.33
Number Concurrent Users        =        300
Model Type                     = INFinite Capacity, NO Resource Queues
Local WAN RoundTrip Time ms          =          20

KTA Message Request bytes            =         100
KTA Message Response bytes           =     1000000
KTA Think Time secs                  =    10.00000
   KTA Txn Local WAN Request secs-E          =     0.00003
    KTA Txn LOADBAL Request secs-E            =     0.00000
     KTA Txn WEBFRONTEND Request secs-E        =     0.01000
      KTA Txn MIDDLEW Request secs-E            =     0.02000
       KTA Txn APPLICATION Request secs-E        =     0.10000


          KTA Txn LOCALDATABASE Request secs-AM  =     3.10000


       KTA Txn APPLICATION Response secs-E       =     0.10000
      KTA Txn MIDDLEW Response secs-E           =     0.02000
     KTA Txn WEBFRONTEND Response secs-E       =     0.01000
    KTA Txn LOADBAL Response secs-E           =     0.00000
   KTA Txn Local WAN Response secs=AC        =     0.30518

Simulation results:
KTA total txns                 =     438900
KTA Simulation minutes         =     333.20
KTA Txns per minute            =    1317.21
KTA Average Response secs      =       3.67
Total Txns                     =     438900
```

Hybrid Local Datacenter, Cloud Database Simulation

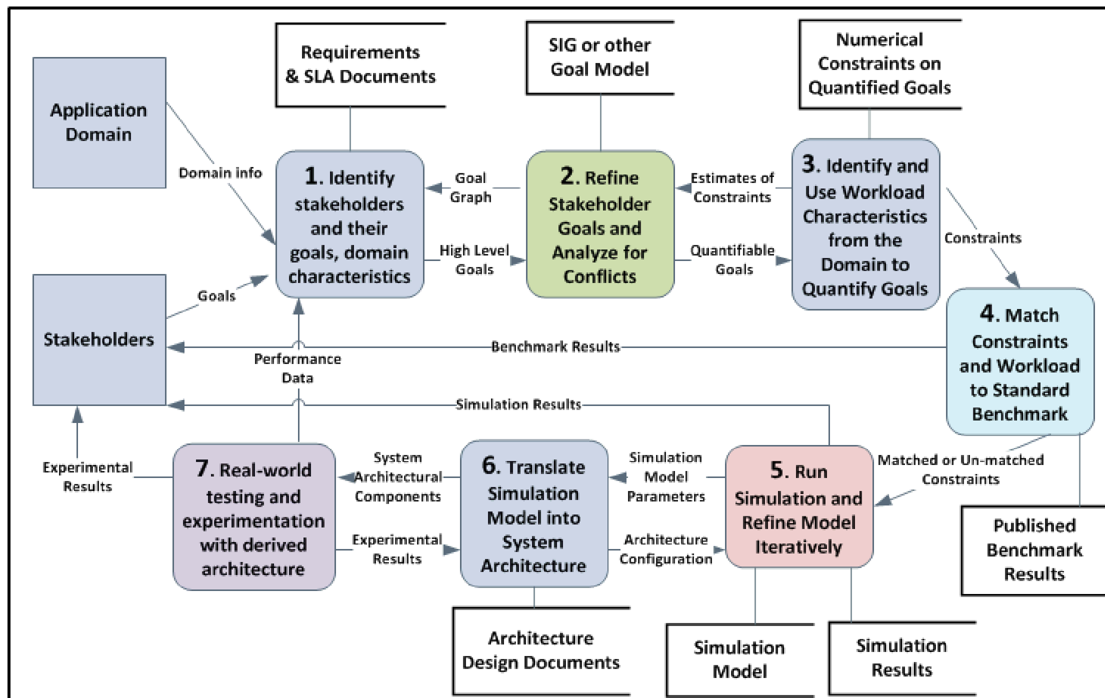
```
MaxCompletions                 =     800000
MaxrunTime minutes             =     333.33
Number Concurrent Users        =        300
Model Type                     = INFinite Capacity, NO Resource Queues
Local WAN RoundTrip Time ms          =          20
Internet Cloud RoundTrip Time ms =              55
KTA Message Request bytes            =         100
KTA Message Response bytes           =     1000000
KTA Think Time secs                  =    10.00000
   KTA Txn Local WAN Request secs-E          =     0.00003
    KTA Txn LOADBAL Request secs-E            =     0.00000
     KTA Txn WEBFRONTEND Request secs-E        =     0.01000
      KTA Txn MIDDLEW Request secs-E            =     0.02000
       KTA Txn APPLICATION Request secs-E        =     0.10000
        KTA Txn CLOUDINTERNET Request secs-AC  =     0.00003
         KTA Txn CLOUDFRONTEND Request secs-E   =     0.13000
          KTA Txn CLOUDDATABASE Request secs-AM  =     3.10000
         KTA Txn CLOUDFRONTEND Response secs-E  =     0.13000
        KTA Txn CLOUDINTERNET Response secs-AC =     0.83923
       KTA Txn APPLICATION Response secs-E       =     0.10000
      KTA Txn MIDDLEW Response secs-E           =     0.02000
     KTA Txn WEBFRONTEND Response secs-E       =     0.01000
    KTA Txn LOADBAL Response secs-E           =     0.00000
   KTA Txn Local WAN Response secs=AC        =     0.30518

Simulation results:
KTA total txns                 =     406200
KTA Simulation minutes         =     333.18
KTA Txns per minute            =    1219.14
KTA Average Response secs      =       4.76
Total Txns                     =     406200
```

Outline

- Motivation
- Research Problem
- Related Work
- The Proposed Solution
  - GoBench
  - GoSim
- Case Studies
- Conclusion

Summary – Integrated Framework Contribution

- The **GoBench and GoSim integrated software engineering framework** demonstrates promise as a vehicle to integrate goals, application workload and architecture infrastructure
- The framework views the simulation model as an architecture-domain-specific **case of knowledge management**
  - The XML, developed to describe the simulation experiment, provides a detailed language to reason about goals, workload and architecture infrastructure
  - A discrete event simulator can be used as a tool to reason about these three important architecture elements



Contributions - Specific

- **In addition to the development of the GoBench GoSim integrated software engineering framework:**
  1. **Stakeholder NFR-goals - Softgoal interdependency graphs (SIG) were used to elicit and document stakeholder performance goals as described by the TPC-C benchmark standard. The SIG provided a more structured approach (more structured than text) to express SLAs and record the rationale for decisions of architectural alternatives**
  2. **OLTP benchmarks for cloud architectures - Java (2,100 lines of code) and Python (4,100 lines of code) versions of the TPC-C benchmark programs were coded, tested and executed in the Google Cloud. Sixteen new cloud TPC-C benchmark result reports (new highest cloud throughput of 7028.7 transactions per minute for 6,000 concurrent users) were documented**
  3. **Architecture resource elasticity - The case experiment discovered Google CloudSQL database limits of elasticity (12 GAE Frontend instances to 16 GAE Frontend instances) through benchmarking. Additionally, the benchmark proved automatic elasticity (628 GAE Frontend instances) for the Google App Engine when using Datastore as a database. The limit of 628 GAE Frontend instances was not a limit of the Google cloud infrastructure. The 628 limit was imposed by a client network security appliance**

Contributions - Specific

- **In addition to the development of the GoBench GoSim integrated software engineering framework:**

4. **Describe a discrete event simulation - Nine cloud simulation result reports were documented that closely align with like benchmarks to increase confidence in the fidelity of the simulation model. A Google App Script (1,100 lines of code) graphical user interface was created to describe a simulation experiment and generate a XML experiment description to be used by multiple discrete event simulators (A SimPy, open source simulation framework, was implemented and executed to provide all simulation results) . The GUI reduced the difficulty in describing a simulation experiment**
5. **Basic software engineering artifacts - Key XML data structures with data elements were built to describe simulation experiments. The XML tag names and values emphasize essential goal, application workload and architecture infrastructure characteristics for continued system maintenance during the operational life of an application (2 performance goal data elements, 39 application workload characteristics data elements and 37 data elements used to describe the architecture infrastructure topology)**

Future Work

Additional non-cloud architecture validation of the GoBench GoSim integrated software engineering framework:

1. **Stakeholder NFR-goals** - Add features to the RE Design tool to generate goal XML for automatic simulation input
2. **OLTP benchmarks for cloud architectures** - Build a local Requirements Engineering Cloud Benchmarking and Simulation Laboratory. Design local lab test-bed to benchmark and simulate a hybrid mix of datacenter and cloud. Extend the Silverlining web site to include benchmark results and open source simulation models
3. **Architecture resource elasticity** - Locate the next Cloud GAE/CloudSQL elasticity constraint above 640 concurrent users by benchmarking the expected new Google CloudSQL database simultaneous connection limit of 3,200. Find the next Cloud GAE/Datastore elasticity constraint above 6,000 concurrent users by using non-UTD Computer Science client network resources
4. **Describe a discrete event simulation** - Augment the capabilities of the Simulation/Forecaster GUI to drag-and-drop graphic elements and animation when creating architecture topology descriptions and simulation execution status demonstration, respectively. Investigate the addition of a "distance to reality" fidelity score.
5. **Basic software engineering artifacts** - Create a local test-bed to compare genetic algorithm results to benchmarks and simulation results. Add local lab cloud emulation (to benchmark and collect resource usage metrics) for Google, Amazon, Azure and OpenStack cloud-provider tests

Publications

- Chung, T. Hill, and N. Subramanian. Silverlining: A Cloud Forecaster Using Benchmarking and Simulation, presented at the *26th Annual IEEE Software Technology Conference, Long Beach, California, March-April, 2014.*
- L. Chung, T. Hill, O. Legunsen, Z. Sun, A. Dsouza and S. Supakkul. A goal-oriented simulation approach for obtaining good private cloud-based system architectures, Original Research Article Journal of Systems and Software, Volume 86, Issue 9, pages 2242-2262, September 2013.
- T. Hill. Software Maintenance and Operations Hybrid Model: An IT Services Industry Architecture Simulation Model Approach, IEEE Research Challenges in Information Science (RCIS), 2011 Fifth International Conference, May 2011.
- T. Hill, S. Supakkul, and L. Chung. Run-time monitoring of system performance: A goal-oriented and system architecture Simulation approach, Requirements@Run.Time, 2010 First International Workshop, Sydney, Australia, pages 31-40, 2010.
- S. Supakkul, T. Hill, E. A. Oladimeji, and L. Chung; “Capturing, Organizing, and Reusing Knowledge of NFRs: An NFR Pattern Approach.” In Proc. 2nd Intl. Workshop on Managing Requirements Knowledge (MaRK'09) in conjunction with RE'09, Atlanta, Sept. 1, 2009.
- S. Supakkul, T. Hill, E. A. Oladimeji, and L. Chung; “Security Threat and Vulnerability Mitigation Patterns: A Case of Credit Card Theft Mitigation.” In Proc. of the 16th Patterns Languages of Programs, Chicago, August 2009.
- T. Hill, S. Supakkul and L. Chung. Confirming and reconfirming architectural decisions on scalability: a goal-driven simulation approach, International Workshop on System/Software Architectures, IWSSA’09, Springer LNCS 5872, 2009.

Questions ?

Thanks, Tom

Supplemental - Publication

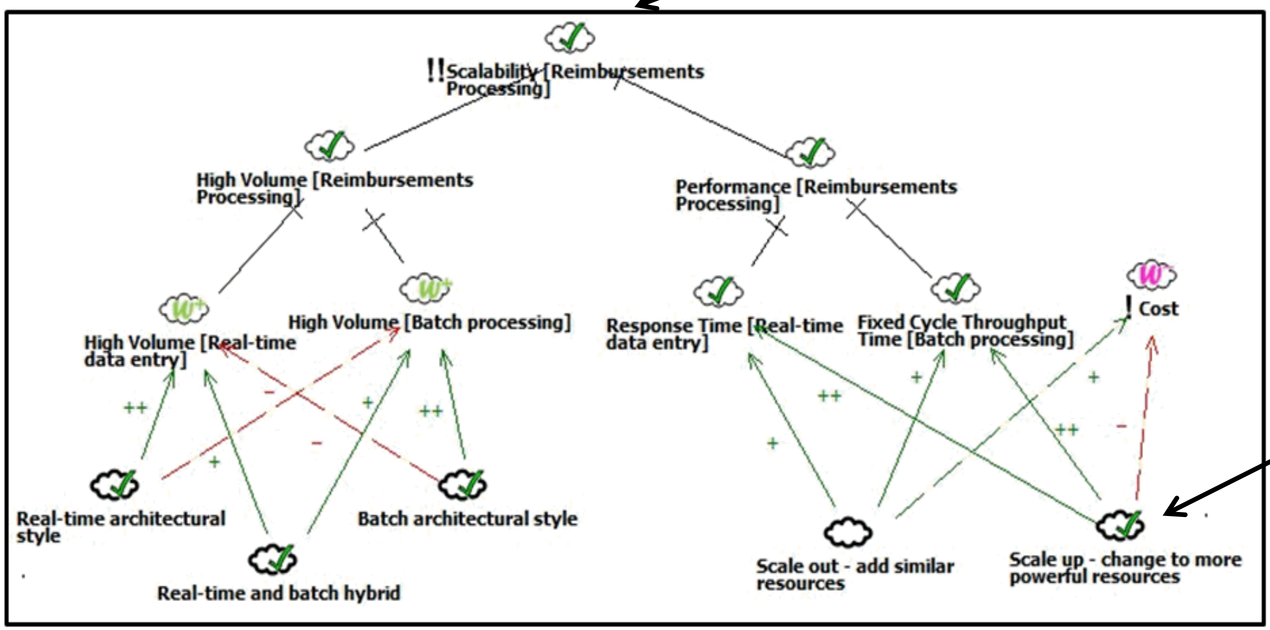
Confirming and Reconfirming Architectural Decisions on Scalability (IWSSA'09, Springer LNCS 5872, 2009)

Approach: Propose an integration of goal-orientation, which is qualitative in nature, and simulation, which is quantitative in nature

Challenge: Difficult to analyze if an architectural design incorporates good decisions or even bad ones

Solution: Use SIG to document NFR scalability goals and sub-goals

Scalability is noted as the primary system goal



Architect decision to select the Scale up option is documented along with the rational

Supplemental - Publication

Run-time monitoring of system performance: A goal-oriented and system architecture Simulation approach

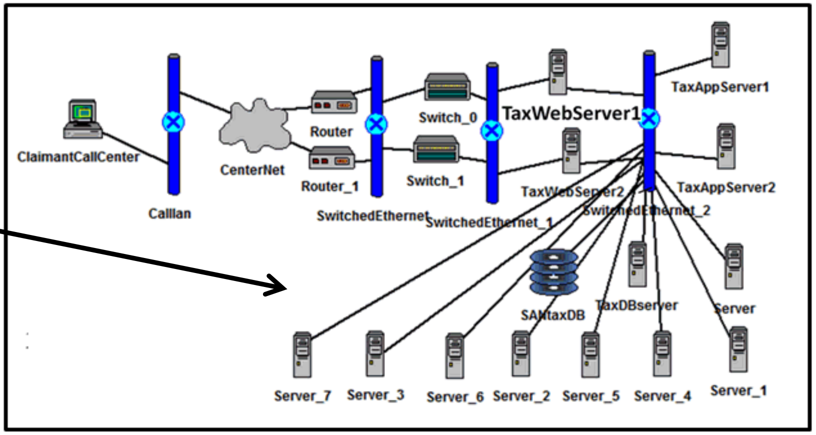
(Requirements@Run.Time, 2010 First International Workshop, pp. 31-40. Sydney, Australia, 2010)

Approach: Propose a goal-oriented framework to record goals, and a system architecture simulation approach to realize and monitor the run-time performance characteristics of the system

Challenge: Simulation models were constructed and used in design and simply ignored during run-time

Solution: A simulation model is constructed and experiments analyzed to consider varying workloads, resource consumptions, and run-time capacities

The production run-time infrastructure (with performance characteristics and capacities) is synchronized with model



Topology of the run-time system is duplicated completely in a simulation model

Supplemental - Publication

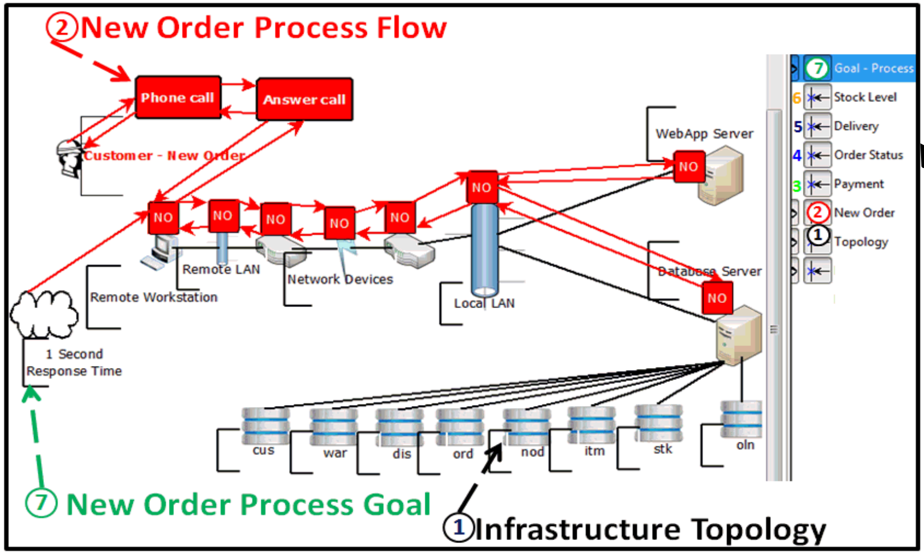
Software Maintenance and Operations Hybrid Model: An IT Services Industry Architecture Simulation Model Approach

(IEEE Research Challenges in Information Science (RCIS), 2011 Fifth International Conference, May 2011)

Approach: Propose an architecture simulation model hybrid, built from existing software development artifacts and operations artifacts, which can endure for the operational life of a system

Challenge: Software maintenance artifacts and operations artifacts continue to diverge down two separate paths filled with duplication and unused information

Solution: A multi-layer simulation model combining goals, process, architecture



1. **A layered infrastructure topology diagram reproduced for input to simulation**
2. **New Order transaction workload and resource usage defined**
3. **3.-6 Other transactions defined**
7. **New Order transaction response time goal is specified**

Supplemental - Publication

A goal-oriented simulation approach for obtaining good private cloud-based system architectures

(Journal of Systems and Software, 86(9): 2242-2262 2013)

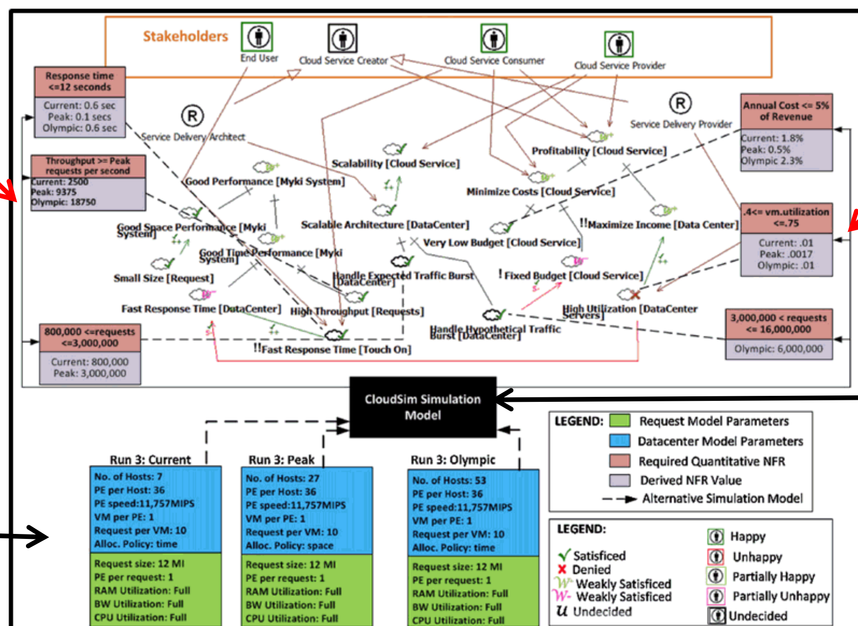
Approach: Propose a goal-oriented simulation approach for cloud-based system design for multiple stakeholders: end user, cloud service customer, provider

Challenge: A lack of methodologies for incorporating stakeholder goals into the design process for such systems, and for assuring with higher confidence

Solution: Simulations are run against various configurations of the model as a way of rationally exploring, evaluating and selecting among incrementally better architectural alternatives

Simulation results: response time, throughput..

Three [current, peak, Olympic] simulation models shown: # hosts, PE/host, PE speed, requests..



Simulation results for each model: annual cost of revenue, VM utilization, hypothetical traffic..

Softgoals (Softgoal Interdependency Graph)**, workflow and architecture integrated graphically via CloudSim** (cloud simulator)

Supplemental - Publication

Google App Engine: Software Benchmark and GAE Simulation Forecaster Grant - Project Summary

(Google App Engine Research Awards, 11/6/2013)

Approach: Build a TPC-C online transaction processing benchmark in the Google cloud using Java and Python

Challenge: Comparing the benchmark performance and cost data points to simulation forecaster results

Solution: A summary of nine project milestones and accompanying metrics (10/2012-10/2013) reported to show the cloud benchmark performance and cost data points along with early simulation results

Milestone Event	Dates	Metrics/Information
Create a statistical model of the TPC-C benchmark databases and transaction workloads	10/2012	14 databases, 5 transactions
Organize Silverlining research team and deploy GAE tutorial programs locally and remotely	11/2012	21 UTD software engineering students
Design , code [Java] and test the complete TPC-C benchmark for GAE and CloudSQL	12/2012-1/2013	2 programs, 2.1 Kloc, 3 person months
Execute TPC-C benchmarks for GAE and CloudSQL	2/2013-5/2013	29 benchmark runs and analysis
Design , code [Python] and test the complete TPC-C benchmark and DMS for GAE Datastore NoSQL	10/2012-6/2013	13 programs, 4.1 Kloc, 9 person months
Begin modification of three open source discrete event simulators for GAE Simulation/forecaster	5/2013	CloudSim, Omnet++, SimPy
Design , code [Google App Script/XML] and test the graphical user interface XML generator	6/2013-7/2013	2 programs, 1.1 Kloc, 3 person months
Generate and import a TPC-C benchmark 750-warehouse-database to the GAE Datastore NoSQL	9/2013	1.3 terabytes
Execute TPC-C benchmarks for GAE Datastore NoSQL	7/2013-10/2013	11 benchmark runs and analysis

GAE Research UTD Project Milestone Table

Supplemental – Reports/Presentations

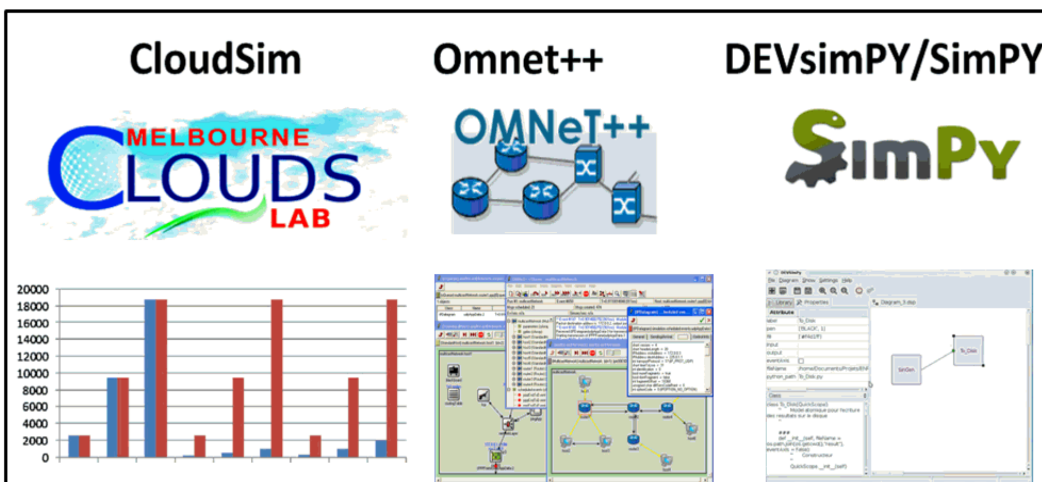
Systems of Systems Engineering: A Goal-driven Architecture Simulation Approach, Quarterly Status Reports and Summaries

(NSF IUCRC Net-Centric Software & Systems Consortium, 2010 – 2014)

Approach: Design a framework to use goals and simulation to help document complex systems-of-systems architectures

Challenge: SoS failures are “… traceable to excessive complexity, poor architectural choices, ill-defined processes, non-validated systems engineering practices or lack of experience in applying valid practices.” [INCOSE Systems Engineering Vision 2020]

Solution: A framework and tools developed to use goals and simulation to understand the behavior complex systems-of-systems architectures



Three open source discrete event simulators used as alternatives to prove the simulation model can confirm Systems of Systems architecture performance and cost:

1. CloudSim – Java

2. Omnet++ - C++

3. SimPY - Python