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
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.
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]

+ 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


- 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


+ 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]
+ 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

1. Identify stakeholders and their goals, domain characteristics
2. Refine Stakeholder Goals and Analyze for Conflicts
3. Identify and Use Workload Characteristics from the Domain to Quantify Goals
4. Match Constraints and Workload to Standard Benchmark
5. Run Simulation and Refine Model Iteratively
6. Translate Simulation Model into System Architecture
7. Real-world testing and experimentation with derived architecture

The function \( f(g_i, w_f, bw_f, ba) \rightarrow (p_m, p_c) \) describes GoBench. Where:
- \( g_i \) the input performance goals
- \( w_f \) the input enterprise workloads
- \( bw_f \) the input benchmark workloads
- \( ba \) the input benchmark architecture
- \( p_m \) the output performance metrics
- \( p_c \) the output performance costs
GoBench Softgoal Interdependency Graph

Step 2 Non-Functional Requirements Performance Goals

Response time  Throughput  Scalability

Quality [Customer-order-system]

Response-time [Customer-order-system]

Throughput [Customer-order-system]

Scalability [Customer-order-system]

Less than 3 seconds response time [Customer-order-system] eq/ 300 transactions per minute [Customer-order-system] eq/ Supporting at least 320 concurrent users [Customer-order-system] Supporting at least 32 warehouses [Customer-order-system]

Response time  Throughput  Scalability
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
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

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

<table>
<thead>
<tr>
<th>Number of Concurrent Users</th>
<th>Benchmark CloudSQL DB Instance D1 (0.5 GB Ram, $0.10/ Hour) tpmC</th>
<th>CloudSQL Maximum Number of FrontEnd Instances (F1 $0.08/ Hour/ Instance) 128MB MEM, 600MHZ CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 users</td>
<td>12.5</td>
<td>1.0</td>
</tr>
<tr>
<td>40 users</td>
<td>50.4</td>
<td>3.0</td>
</tr>
<tr>
<td>320 users</td>
<td>338.1</td>
<td>12.0</td>
</tr>
<tr>
<td>640 users</td>
<td>3</td>
<td>16.0</td>
</tr>
</tbody>
</table>
GoBench TPC-C Benchmark(s) Results
Step 4 Google Cloud Results for 15 Benchmark Experiments

Benchmark throughput in transactions per minute (tpmC)

The number of concurrent users, generating transactions

Number of Cloud FrontEnd instances automatically allocated
GoBench TPC-C Benchmark(s) Results

Step 4 Google Cloud Results for 16 Benchmark Experiments Table

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

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

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

1. Identify stakeholders and their goals, domain characteristics
2. Refine Stakeholder Goals and Analyze for Conflicts
3. Identify and Use Workload Characteristics from the Domain to Quantify Goals
4. Match Constraints and Workload to Standard Benchmark
5. Run Simulation and Refine Model Iteratively
6. Translate Simulation Model into System Architecture
7. Real-world testing and experimentation with derived architecture

Step 5 (Run Simulation Experiments GoSim Function) Highlighted

The function \( f(g, W, a) \rightarrow (p_m, p_c) \) describes GoSim. Where:
- \( g \) the input performance goals
- \( W \) the input enterprise workloads
- \( a \) the input resource architecture
- \( p_m \) the output performance metrics
- \( p_c \) the output performance costs
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

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

Performance Goals

```
<xml version="1.0"?>
<performancegoals>
  <!-- Performance goals -->
</performancegoals>
```

Application Workload

```
<applicationworkload>
  <!-- Application workload -->
</applicationworkload>
```

Architecture Infrastructure

```
<architectureinfrastructure>
  <!-- Architecture infrastructure -->
</architectureinfrastructure>
```
**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**

**Workload requests represent resource usage and architecture infrastructure components represent capacity**

**Simulator generates workload of multiple users and collects metrics**

### 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

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

### 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

![Diagram of simulation process](image-url)
GoSim Execute Simulation Forecaster

Google Cloud Project Simulation Forecaster Mean Interarrival Algorithm Example

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
“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

<table>
<thead>
<tr>
<th>Simulation-run-title</th>
<th>Run-date-time</th>
<th>Latency-goal</th>
<th>Throughput-goal</th>
<th>App-group</th>
<th>smtwtfs</th>
<th>Op-hours</th>
<th>#-users</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST for New 4.0 GB D1 DB Server</td>
<td>2014-08-02 15:30:52</td>
<td>2 secs</td>
<td>300 tpm TPCC-Benchmark</td>
<td>yyyy.yy</td>
<td>24</td>
<td>320</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>App-txn-title</th>
<th>SIM-MINUTES</th>
<th>SIM-AVG-LATENCY-SECS</th>
<th>SIM-THROUGHPUT-PER-MIN</th>
<th>SIM-TXN-COUNT</th>
<th>Txn-workload-%</th>
</tr>
</thead>
<tbody>
<tr>
<td>New-Order</td>
<td>333.33</td>
<td>0.00</td>
<td>378.58</td>
<td>126194</td>
<td>45</td>
</tr>
<tr>
<td>Payment</td>
<td>333.33</td>
<td>0.00</td>
<td>444.86</td>
<td>148204</td>
<td>43</td>
</tr>
<tr>
<td>Delivery</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Order-Status</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Stock-Level</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

II. **Simulation-forecast-of-GAE-frontend-variable-resource-usage**

<table>
<thead>
<tr>
<th>Used</th>
<th>Charge ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Daily Instance F1 Hours [average 10 instance 24 hours ea $ 0.08/Hour]</td>
<td>240</td>
</tr>
<tr>
<td>2. Daily Bandwidth Out average Gigabytes[$ 0.12/Gigabyte]</td>
<td>1.668</td>
</tr>
<tr>
<td>3. 30-day Month Total Estimate</td>
<td>582.00</td>
</tr>
</tbody>
</table>

III. **Simulation-forecast-of-CloudSQL-variable-resource-usage**

<table>
<thead>
<tr>
<th>Debits ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Daily SQL Service D1 Usage Hours [average 24 hours, $ 0.10/Hour]</td>
</tr>
<tr>
<td>2. Daily SQL Service Read and Write Count [average 35.2 million RWs, $0.10/million]</td>
</tr>
<tr>
<td>3. Daily SQL Service Disk Usage [average 4.00 GB for the month, $ 0.24/GB/month]</td>
</tr>
<tr>
<td>4. 30-day Month Total Estimate</td>
</tr>
</tbody>
</table>

**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

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

1. Google Cloud Platform Abstraction Layer
2. Silver lining PaaS Abstraction Layer
3. Mapping Layer
4. Infrastructure Abstraction Hardware/software Layer
5. Performance Implementation Hardware Layer

- Processor: Intel core 2 dual core, 2.93GHz, 27,079 MIPs, F1—F4, 1G 600MHz—2.4GHz, B1—B8, 600MHz—4.8GHz
- Memory RAM: 0.5—16GB
- Disk: Capacity 300GB—420GB

MySQL - big tables log(rows 600,000) log(index block bytes 1024/3*2/(index bytes 3 + pointer bytes 4)) + 1 = 4 seeks/read, 6 seeks/write

3. Application instruction path length aggregation [MIPs] can be matched with processor capacity [MIPs]

Average seek – read/write ms
3.6/4.2 = 15.5/15.5. Average seek 10ms
Google Cloud Database Configuration Alternatives

Database CloudSQL and Datastore:

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

   - OLTP Database Model
   - Internet
   - Requests → Response Messages
   - Google App Engine Frontend
   - SQL Request Statements → SQL Response Messages
   - Google CloudSQL Database Engine Service
     - 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

   - OLTP Web Datastore Cloud NoSQL Model
   - Internet
   - Requests → Response Messages
   - Google App Engine Frontend
   - Datastore
     - Write
     - Read

   - 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)

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

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

```sql
USE tpcc32;
-- Table structure for table 'customer'
--
DROP TABLE IF EXISTS 'customer';
/*140101 SET @saved_cs_client = @character_set_client */;
/*140101 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_store_city` 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_TIMESTAMP,
  `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;
/*140101 SET character_set_client = @saved_cs_client */;
--
-- Dumping data for table 'customer'
--
--
LOCK TABLES `customer` WRITE;
/*1400000 ALTER TABLE `customer` DISABLE KEYS */;
INSERT INTO `customer` VALUES (1,111,0.4020,0.4,0.418,0.4,0.421,0.4,0.424,0.4,0.427,0.4,0.430,0.4,0.433,0.4,0.436,0.4,0.439,0.4,0.442,0.4,0.445,0.4,0.448,0.4,0.451,0.4,0.454,0.4,0.457,0.4,0.460,0.4,0.463,0.4,0.466,0.4,0.469,0.4,0.472,0.4,0.475,0.4,0.478,0.4,0.481,0.4,0.484,0.4,0.487,0.4,0.490,0.4,0.493,0.4,0.496,0.4,0.499,0.4,0.502,0.4,0.505,0.4,0.508,0.4,0.511,0.4,0.514,0.4,0.517,0.4,0.520,0.4,0.523,0.4,0.526,0.4,0.529,0.4,0.532,0.4,0.535,0.4,0.538,0.4,0.541,0.4,0.544,0.4,0.547,0.4,0.550,0.4,0.553,0.4,0.556,0.4,0.559,0.4,0.562,0.4,0.565,0.4,0.568,0.4,0.571,0.4,0.574,0.4,0.577,0.4,0.580,0.4,0.583,0.4,0.586,0.4,0.589,0.4,0.592,0.4,0.595,0.4,0.598,0.4,0.601,0.4,0.604,0.4,0.607,0.4,0.610,0.4,0.613,0.4,0.616,0.4,0.619,0.4,0.622,0.4,0.625,0.4,0.628,0.4,0.631,0.4,0.634,0.4,0.637,0.4,0.640,0.4,0.643,0.4,0.646,0.4,0.649,0.4,0.652,0.4,0.655,0.4,0.658,0.4,0.661,0.4,0.664,0.4,0.667,0.4,0.670,0.4,0.673,0.4,0.676,0.4,0.679,0.4,0.682,0.4,0.685,0.4,0.688,0.4,0.691,0.4,0.694,0.4,0.697,0.4,0.700,0.4,0.703,0.4,0.706,0.4,0.709,0.4,0.712,0.4,0.715,0.4,0.718,0.4,0.721,0.4,0.724,0.4,0.727,0.4,0.730,0.4,0.733,0.4,0.736,0.4,0.739,0.4,0.742,0.4,0.745,0.4,0.748,0.4,0.751,0.4,0.754,0.4,0.757,0.4,0.760,0.4,0.763,0.4,0.766,0.4,0.769,0.4,0.772,0.4,0.775,0.4,0.778,0.4,0.781,0.4,0.784,0.4,0.787,0.4,0.790,0.4,0.793,0.4,0.796,0.4,0.799,0.4,0.802,0.4,0.805,0.4,0.808,0.4,0.811,0.4,0.814,0.4,0.817,0.4,0.820,0.4,0.823,0.4,0.826,0.4,0.829,0.4,0.832,0.4,0.835,0.4,0.838,0.4,0.841,0.4,0.844,0.4,0.847,0.4,0.850,0.4,0.853,0.4,0.856,0.4,0.859,0.4,0.862,0.4,0.865,0.4,0.868,0.4,0.871,0.4,0.874,0.4,0.877,0.4,0.880,0.4,0.883,0.4,0.886,0.4,0.889,0.4,0.892,0.4,0.895,0.4,0.898,0.4,0.901,0.4,0.904,0.4,0.907,0.4,0.910,0.4,0.913,0.4,0.916,0.4,0.919,0.4,0.922,0.4,0.925,0.4,0.928,0.4,0.931,0.4,0.934,0.4,0.937,0.4,0.940,0.4,0.943,0.4,0.946,0.4,0.949,0.4,0.952,0.4,0.955,0.4,0.958,0.4,0.961,0.4,0.964,0.4,0.967,0.4,0.970,0.4,0.973,0.4,0.976,0.4,0.979,0.4,0.982,0.4,0.985,0.4,0.988,0.4,0.991,0.4,0.994,0.4,0.997,0.4,0.998,0.4,0.999,0.4)
*/
```
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
- Response message size in bytes: [Image of graph with statistics for users (agents) 1 through 40]
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

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

#### Lab computer name
- Accumulated number of users
- Number of Front End instances assigned by Google

### Table: Resource Usage Data

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Users (count)</th>
<th>Duration (seconds)</th>
<th>WHSE Start</th>
<th>#Users Accum</th>
<th>Duration Minutes</th>
<th>Duration Hours</th>
<th>Actual Start Time</th>
<th># GAE FE Instances</th>
<th># Requests Processed</th>
<th>Bandwidth Download Mbps</th>
<th>Bandwidth Upload Mbps</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>HM1</td>
<td>500</td>
<td>5100</td>
<td>0</td>
<td>500</td>
<td>85</td>
<td>1.42</td>
<td>3:22</td>
<td>255</td>
<td>92,660</td>
<td>56</td>
<td>20</td>
<td>Start of bench</td>
</tr>
<tr>
<td>HM2</td>
<td>500</td>
<td>4980</td>
<td>50</td>
<td>1,000</td>
<td>83</td>
<td>1.38</td>
<td>3:24</td>
<td>322</td>
<td>85,935</td>
<td>61</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>HM3</td>
<td>500</td>
<td>4800</td>
<td>100</td>
<td>1,500</td>
<td>81</td>
<td>1.35</td>
<td>3:26</td>
<td>363</td>
<td>87,215</td>
<td>63</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>HM4</td>
<td>500</td>
<td>4740</td>
<td>150</td>
<td>2,000</td>
<td>79</td>
<td>1.32</td>
<td>3:28</td>
<td>373</td>
<td>84,464</td>
<td>56</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>HM5</td>
<td>500</td>
<td>4620</td>
<td>200</td>
<td>2,500</td>
<td>77</td>
<td>1.28</td>
<td>3:30</td>
<td>445</td>
<td>81,629</td>
<td>65</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>HM6</td>
<td>500</td>
<td>4500</td>
<td>250</td>
<td>3,000</td>
<td>75</td>
<td>1.25</td>
<td>3:32</td>
<td>464</td>
<td>79,392</td>
<td>60</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>HM7</td>
<td>500</td>
<td>4380</td>
<td>300</td>
<td>3,500</td>
<td>73</td>
<td>1.22</td>
<td>3:34</td>
<td>487</td>
<td>76,969</td>
<td>59</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>HM8</td>
<td>500</td>
<td>4260</td>
<td>350</td>
<td>4,000</td>
<td>71</td>
<td>1.18</td>
<td>3:36</td>
<td>512</td>
<td>73,614</td>
<td>60</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>HM9</td>
<td>500</td>
<td>4140</td>
<td>400</td>
<td>4,500</td>
<td>69</td>
<td>1.15</td>
<td>3:38</td>
<td>555</td>
<td>70,932</td>
<td>60</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>HM10</td>
<td>500</td>
<td>4020</td>
<td>450</td>
<td>5,000</td>
<td>67</td>
<td>1.12</td>
<td>3:40</td>
<td>577</td>
<td>68,282</td>
<td>59</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>HM11</td>
<td>500</td>
<td>3900</td>
<td>500</td>
<td>5,500</td>
<td>65</td>
<td>1.08</td>
<td>3:42</td>
<td>605</td>
<td>65,940</td>
<td>56</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>HM12</td>
<td>500</td>
<td>3780</td>
<td>550</td>
<td>6,000</td>
<td>63</td>
<td>1.05</td>
<td>3:44</td>
<td>628</td>
<td>63,348</td>
<td>55</td>
<td>20</td>
<td>Level Instances</td>
</tr>
</tbody>
</table>

6,000 concurrent users level benchmark

- 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**

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

<table>
<thead>
<tr>
<th>Number of Concurrent Users</th>
<th>10 users</th>
<th>40 users</th>
<th>640 users</th>
<th>1,200 users</th>
<th>6,000 users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datastore F4_1G (2400MHZ, 1024MB) tpmC</td>
<td>12.0</td>
<td>49.1</td>
<td>392.3</td>
<td>778.7</td>
<td>1,458.1</td>
</tr>
<tr>
<td>Datastore Maximum Number of Front End Instances (F4_1G $0.48/Hour/Instance) 1024 MB MEM, 2400MHZ CPU</td>
<td>5.0</td>
<td>8.0</td>
<td>55.0</td>
<td>83.0</td>
<td>154.0</td>
</tr>
</tbody>
</table>
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
Excerpt of a Generated XML Description of a Google Cloud Architecture Infrastructure with Component Costs Highlighted

```
<infrastructureconfig>
  <webclienticon>webclienticon.png</webclienticon>
  <numberclients>320</numberclients>
  <requestresponsersec>10</requestresponsersec>
  <requestcapmbpssec>50</requestcapmbpssec>
  <costresponsepergbyte>0.12</costresponsepergbyte>
  <internetcon>inet.png</internetcon>
  <requestresponsersec>10</requestresponsersec>
  <gaefrontendicon>gaefrontend.png</gaefrontendicon>
  <instanceclass>F1</instanceclass>
  <instanceclockmhz>600</instanceclockmhz>
  <instancecbyte>128</instancecbyte>
  <costperinsthour>0.08</costperinsthour>
  <requestresponsersec>10</requestresponsersec>
  <clouddbengineicon>clouddbengine.png</clouddbengineicon>
  <instanceclass>D1</instanceclass>
  <instancecbyte>5</instancecbyte>
  <costperinsthour>0.10</costperinsthour>
  <costpermio>0.10</costpermio>
  <requestresponsersec>10</requestresponsersec>
  <dbstorageicon>dbstorage.png</dbstorageicon>
  <coststoragepergbytepermio>0.24</coststoragepergbytepermio>
  <coststoragepergbytepermonth>0.24</coststoragepergbytepermonth>
</infrastructureconfig>
```

- 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: `coststoragepergbytepermonth`
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 # seconds 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

<table>
<thead>
<tr>
<th>Simulation-run-title</th>
<th>Run-date-time</th>
<th>Latency-goal</th>
<th>Throughput-goal</th>
<th>App-group</th>
<th>sntwfs</th>
<th>Op-hours</th>
<th>#-users</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST for sim</td>
<td>2013-12-31 17:36:42</td>
<td>2 secs</td>
<td>Max tpm TPC-C-Benchmark</td>
<td>yyyyyyy</td>
<td>2</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>App-tax-title</th>
<th>SIM-MINUTES SIM-AVG-LATENCY-SECS</th>
<th>SIM-THROUGHPUT-PER-MIN SIM-TXN-COUNT</th>
<th>Tnx-workload-%</th>
</tr>
</thead>
<tbody>
<tr>
<td>New-Order</td>
<td>333.32</td>
<td>0.00</td>
<td>11.02</td>
</tr>
<tr>
<td>Payment</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Delivery</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Order-Status</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Stock-Level</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### II. Simulation-forecast-of-GAE-frontend-variable-resource-usage

<table>
<thead>
<tr>
<th>USED</th>
<th>CHARGE($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Daily F4_10 2 instances</td>
<td>$24</td>
</tr>
<tr>
<td>2. Daily Bandwidth Out average Gigabytes</td>
<td>$0.004</td>
</tr>
<tr>
<td>3. 30-day Month Total Estimate</td>
<td>$22,156.53</td>
</tr>
</tbody>
</table>

#### III. Simulation-forecast-of-Datastore-variable-resource-usage

<table>
<thead>
<tr>
<th>USED</th>
<th>CHARGE($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Daily Datastore Writes per million operations</td>
<td>$0.31</td>
</tr>
<tr>
<td>2. Daily Datastore Reads per million operations</td>
<td>$0.06</td>
</tr>
<tr>
<td>3. Daily Datastore Storage per GB per day</td>
<td>$1350.00</td>
</tr>
<tr>
<td>4. 30-day Month Total Estimate</td>
<td>$6002.67</td>
</tr>
</tbody>
</table>

### New Order Application Throughput

- **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

**Number of concurrent Users**: 6,000
Google Cloud Benchmark(s) and Simulation(s) Fidelity

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

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

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].**

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

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxCompletions</td>
<td>800000</td>
</tr>
<tr>
<td>MaxrunTime minutes</td>
<td>333.33</td>
</tr>
<tr>
<td>Number Concurrent Users</td>
<td>300</td>
</tr>
<tr>
<td>Model Type</td>
<td>Infinite Capacity, No Resource Queues</td>
</tr>
<tr>
<td>Local WAN RoundTrip Time ms</td>
<td>20</td>
</tr>
<tr>
<td>KTA Message Request bytes</td>
<td>100</td>
</tr>
<tr>
<td>KTA Message Response bytes</td>
<td>1000000</td>
</tr>
<tr>
<td>KTA Think Time secs</td>
<td>10.00000</td>
</tr>
<tr>
<td>KTA Txn Local WAN Request secs-E</td>
<td>0.0003</td>
</tr>
<tr>
<td>KTA Txn LOADBAL Request secs-E</td>
<td>0.00000</td>
</tr>
<tr>
<td>KTA Txn WEBFRONTEND Request secs-E</td>
<td>0.01000</td>
</tr>
<tr>
<td>KTA Txn MIDDLEWE Request secs-E</td>
<td>0.02000</td>
</tr>
<tr>
<td>KTA Txn APPLICATION Request secs-E</td>
<td>0.10000</td>
</tr>
<tr>
<td>KTA Txn LOCALDATABASE Request secs-AM</td>
<td>3.10000</td>
</tr>
<tr>
<td>KTA Txn APPLICATION Response secs-E</td>
<td>0.10000</td>
</tr>
<tr>
<td>KTA Txn MIDDLEWE Response secs-E</td>
<td>0.02000</td>
</tr>
<tr>
<td>KTA Txn WEBFRONTEND Response secs-E</td>
<td>0.01000</td>
</tr>
<tr>
<td>KTA Txn LOADBAL Response secs-AC</td>
<td>0.00000</td>
</tr>
<tr>
<td>KTA Tnx Local WAN Response secs-AC</td>
<td>0.30516</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxCompletions</td>
<td>800000</td>
</tr>
<tr>
<td>MaxrunTime minutes</td>
<td>333.33</td>
</tr>
<tr>
<td>Number Concurrent Users</td>
<td>300</td>
</tr>
<tr>
<td>Model Type</td>
<td>Infinite Capacity, No Resource Queues</td>
</tr>
<tr>
<td>Local WAN RoundTrip Time ms</td>
<td>20</td>
</tr>
<tr>
<td>Internet Cloud RoundTrip Time ms</td>
<td>55</td>
</tr>
<tr>
<td>KTA Message Request bytes</td>
<td>100</td>
</tr>
<tr>
<td>KTA Message Response bytes</td>
<td>1000000</td>
</tr>
<tr>
<td>KTA Think Time secs</td>
<td>10.00000</td>
</tr>
<tr>
<td>KTA Txn Local WAN Request secs-E</td>
<td>0.0003</td>
</tr>
<tr>
<td>KTA Txn LOADBAL Request secs-E</td>
<td>0.00000</td>
</tr>
<tr>
<td>KTA Txn WEBFRONTEND Request secs-E</td>
<td>0.01000</td>
</tr>
<tr>
<td>KTA Txn MIDDLEWE Request secs-E</td>
<td>0.02000</td>
</tr>
<tr>
<td>KTA Txn APPLICATION Request secs-E</td>
<td>0.10000</td>
</tr>
<tr>
<td>KTA Txn APPLICATION Response secs-E</td>
<td>0.10000</td>
</tr>
<tr>
<td>KTA Txn MIDDLEWE Response secs-E</td>
<td>0.02000</td>
</tr>
<tr>
<td>KTA Txn WEBFRONTEND Response secs-E</td>
<td>0.01000</td>
</tr>
<tr>
<td>KTA Tnx LOCALDATABASE Request secs-AM</td>
<td>3.10000</td>
</tr>
<tr>
<td>KTA Tnx CLOUDINTERNET Request secs-AC</td>
<td>0.00003</td>
</tr>
<tr>
<td>KTA Tnx CLOUDFRONTEND Request secs-E</td>
<td>0.13000</td>
</tr>
<tr>
<td>KTA Tnx CLOUDDATABASE Request secs-AM</td>
<td>3.10000</td>
</tr>
<tr>
<td>KTA Tnx CLOUDINTERNET Response secs-AC</td>
<td>0.83923</td>
</tr>
<tr>
<td>KTA Tnx APPLICATION Response secs-E</td>
<td>0.10000</td>
</tr>
<tr>
<td>KTA Tnx MIDDLEWE Response secs-E</td>
<td>0.02000</td>
</tr>
<tr>
<td>KTA Tnx WEBFRONTEND Response secs-E</td>
<td>0.01000</td>
</tr>
<tr>
<td>KTA Tnx LOADBAL Response secs-AC</td>
<td>0.00000</td>
</tr>
<tr>
<td>KTA Tnx Local WAN Response secs-AC</td>
<td>0.30516</td>
</tr>
</tbody>
</table>

Simulation results:

- KTA total txns = 406200
- KTA Simulation minutes = 333.16
- 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.

![Diagram of the integrated framework process]

1. Identify stakeholders and their goals, domain characteristics
2. Refine Stakeholder Goals and Analyze for Conflicts
3. Identify and Use Workload Characteristics from the Domain to Quantify Goals
4. Match Constraints and Workload to Standard Benchmark
5. Run Simulation and Refine Model Iteratively
6. Translate Simulation Model into System Architecture
7. Real-world testing and experimentation with derived architecture

- Numerical Constraints on Quantified Goals
- SIG or other Goal Model
- Requirements & SLA Documents
- Application Domain
- Stakeholders
- System Architectural Components
- Experimental Results
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


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
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.-6 Other transactions defined
7. New Order transaction response time goal is specified
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)
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

<table>
<thead>
<tr>
<th>Milestone Event</th>
<th>Dates</th>
<th>Metrics/Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a statistical model of the TPC-C benchmark databases and transaction workloads</td>
<td>10/2012</td>
<td>14 databases, 5 transactions</td>
</tr>
<tr>
<td>Organize Silverlining research team and deploy GAE tutorial programs locally and remotely</td>
<td>11/2012</td>
<td>21 UTD software engineering students</td>
</tr>
<tr>
<td>Design, code [Java] and test the complete TPC-C benchmark for GAE and CloudSQL</td>
<td>12/2012-1/2013</td>
<td>2 programs, 2.1 Kloc, 3 person months</td>
</tr>
<tr>
<td>Execute TPC-C benchmarks for GAE and CloudSQL</td>
<td>2/2013-5/2013</td>
<td>29 benchmark runs and analysis</td>
</tr>
<tr>
<td>Design, code [Python] and test the complete TPC-C benchmark and DMS for GAE Datastore NoSQL</td>
<td>10/2012-6/2013</td>
<td>13 programs, 4.1 Kloc, 9 person months</td>
</tr>
<tr>
<td>Begin modification of three open source discrete event simulators for GAE Simulation/forecaster</td>
<td>5/2013</td>
<td>CloudSim, Omnet++, SimPy</td>
</tr>
<tr>
<td>Design, code [Google App Script/XML] and test the graphical user interface XML generator</td>
<td>6/2013-7/2013</td>
<td>2 programs, 1.1 Kloc, 3 person months</td>
</tr>
<tr>
<td>Generate and import a TPC-C benchmark 750-warehouse-database to the GAE Datastore NoSQL</td>
<td>9/2013</td>
<td>1.8 terabytes</td>
</tr>
<tr>
<td>Execute TPC-C benchmarks for GAE Datastore NoSQL</td>
<td>7/2013-10/2013</td>
<td>11 benchmark runs and analysis</td>
</tr>
</tbody>
</table>
Supplemental – Reports/Presentations


**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