Agile Performance Engineering with Cloud

By Ken Y. Chan

Timeleap Inc.

Dec 28, 2016

Outline

1. Importance of Performance Engineering
2. Background in Performance Engineering
3. Traditional Server Farms versus Clouds
4. Resources and Skill Sets
5. Approach and Process
6. Guidelines
7. Conclusion

Importance of Performance Engineering

1. Maintain Business SLO's and SLA's with your partners, clients, end users, and consumers; keep them happy!
2. Happy partners, clients, consumers, or users lead to better retention (avoid revenue loss), and indirectly more business and revenue
3. Poor performance is a sign of system degradation and future service outages
4. Performance problems need more time to solve than functional problems

Background in Performance Engineering - 1

Background

- Part of ITIL
- Extensive Academic Research and History (e.g. CMU, MIT, etc)
- Encompasses the set of roles, skills, activities, practices, tools, and deliverables applied at every phase of the SDLC
- A system is designed, implemented, and operationally supported to meet the non-functional requirements for performance
- Model, Simulation, Performance Test, Design, Implementation, and Tuning

Common Performance Metrics

- Availability
- Response time
- Processing speed
- Capacity and Utilization
- Latency
- Bandwidth
- Throughput
- Scalability

Background of Performance Engineering - 2

Why is it so hard?

Deep technical skills in Mathematics and computing science or engineering

Based on Queuing Theory and Statistics

A hard topic and often being neglected

Think of a computing system as a car or as a building

1. Poorly tuned components would shift bottlenecks to other components (analogous to city traffic)
    a. May temporarily hide the real problem(s)
    b. May lead to eventual system breakdown
2. Alternatively, simply increase the entire system footprint
    a. However, random tuning adds unnecessary footprints (costs $$$$ and reduces maintainability)
    b. May temporarily hide the real problem(s)
    c. Harder to tune with larger footprint at a later time
    d. Costs more $$$, time, and effort down the road
    e. No different from a major car or building overhaul
3. The correct approach is to engineer for performance in piece-meals (iterations)

Traditional Server Farms versus Clouds

On-Premise, Co-located, or Hosted

- Some networking gears may be shared
- A customer has a lot of controls on the physical computing resources
- Performance model is based on queuing in networking nodes, computing nodes, and data nodes
- Different types of queuing for different nodes
- Performance is more deterministic with traditional data centres than with clouds

Public, Private, or Hybrid Clouds

- Networking gears are typically multi-tenant
- Physical computing resources are shared, except in some cases for private clouds
- Performance model is based on queuing in network nodes, computing nodes, and data nodes
- Develop a logical performance model by aggregating multi-tenant nodes into one node
- Less deterministic because of other tenants
- Need more model refactoring to mitigate risks with multiple tenants (TBD in my next presentation)

Resources and Skill Sets

Required Resources

- Performance Modelling Tools
- Performance Simulation Tools
- Performance Test Tools
- Performance Profilers or Monitoring Tools

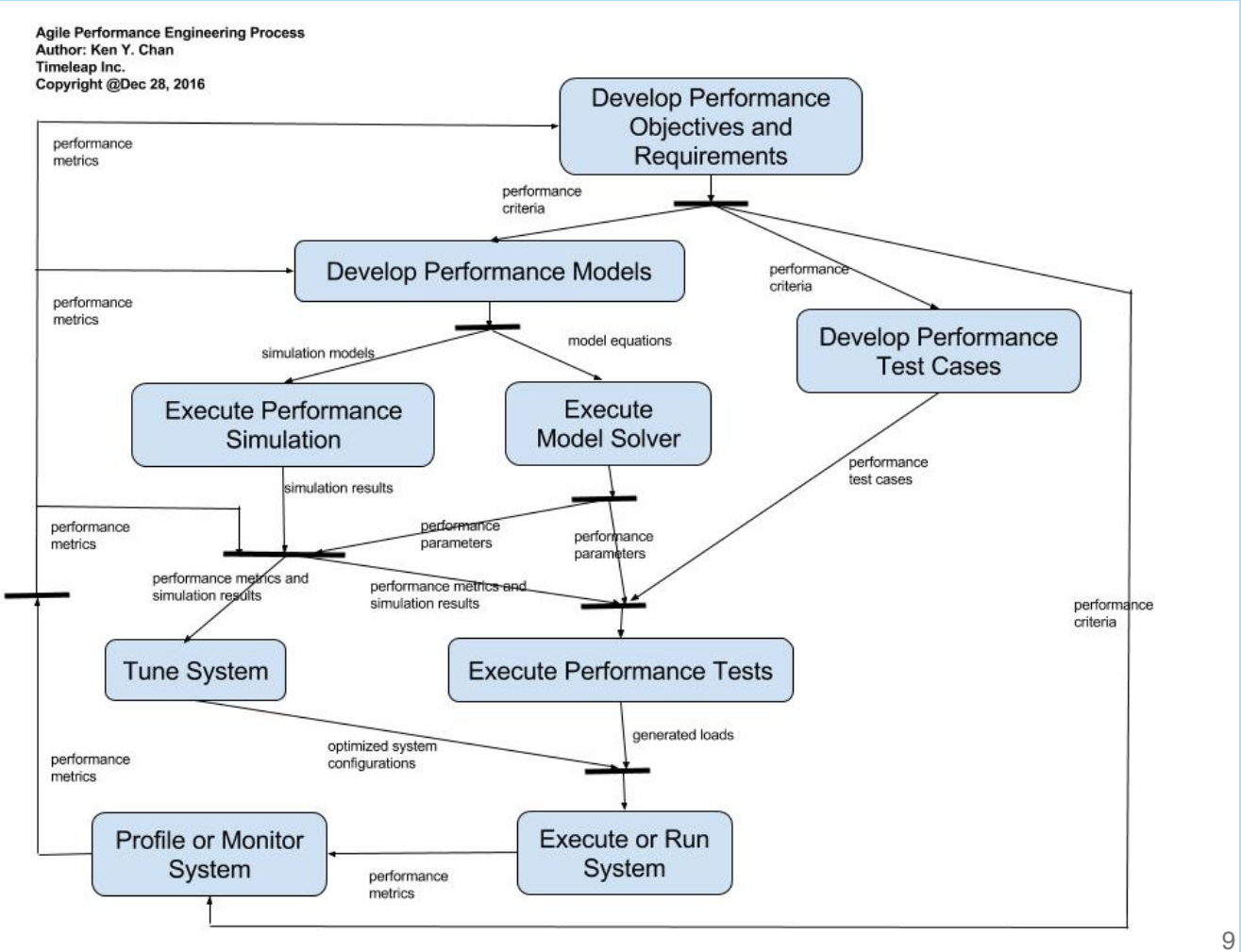
Required Skill Sets

- Queuing Theory and System Performance Engineering
- Theory of Probability and Statistics
- Theory of Computational Complexity
- System Architecture and Design Patterns
- Cloud and Network Architecture
- Database
- Bonus: AI, Big Data (TBD in my next presentation)

Agile Performance Engineering Approach with Cloud

1. Set Initial Performance Objectives for Sprint 1.
2. Develop Performance Models and Test Cases. Abstract Multi-tenant nodes as one node
3. Tune the system configuration based on simulations and metrics
4. Measure Performance Metrics via Performance Monitors
5. Compare actual metrics against performance objectives
6. Revise Performance Objectives for Sprint 2+. Actual metrics and objectives should converge after every sprint

Agile Performance Engineering Process



Develop Performance Model

1. There are a number of Performance Modelling or Simulation Tools: a) SciPy - Python DES, b) R DES, c) Matlab SimEvents, d) JMT, e) Octave, f) and etc.
2. In general, a system consists of multiple stations or nodes (e.g. M/G/k)
3. Each node belongs to a class/type with attributes (e.g. FIFO, LIFO, etc)
4. A class may have attributes: a) arrival distribution, b) processing distribution, c) # of channels, d) size of queue, e) max number of customers
5. May use regression analysis to select the best distributions for your classes
6. Abstract multi-tenant cloud nodes into a single node (TBD in my next presentation)

Simulation versus Solver

1. A Simulation (e.g. M/G/k) generates random discrete events based on assigned distributions, and feeds them as inputs into simulated stations

2. Executing a simulation model typically yields these results at each node: a) queue length, b) queue time, c) response time, d) utilization, e) throughput

3. A Solver (e.g. M/G/k) uses linear or nonlinear equation solving algorithms, or MVA to solve the model as multivariate equations (e.g. response time, utilization, and etc)

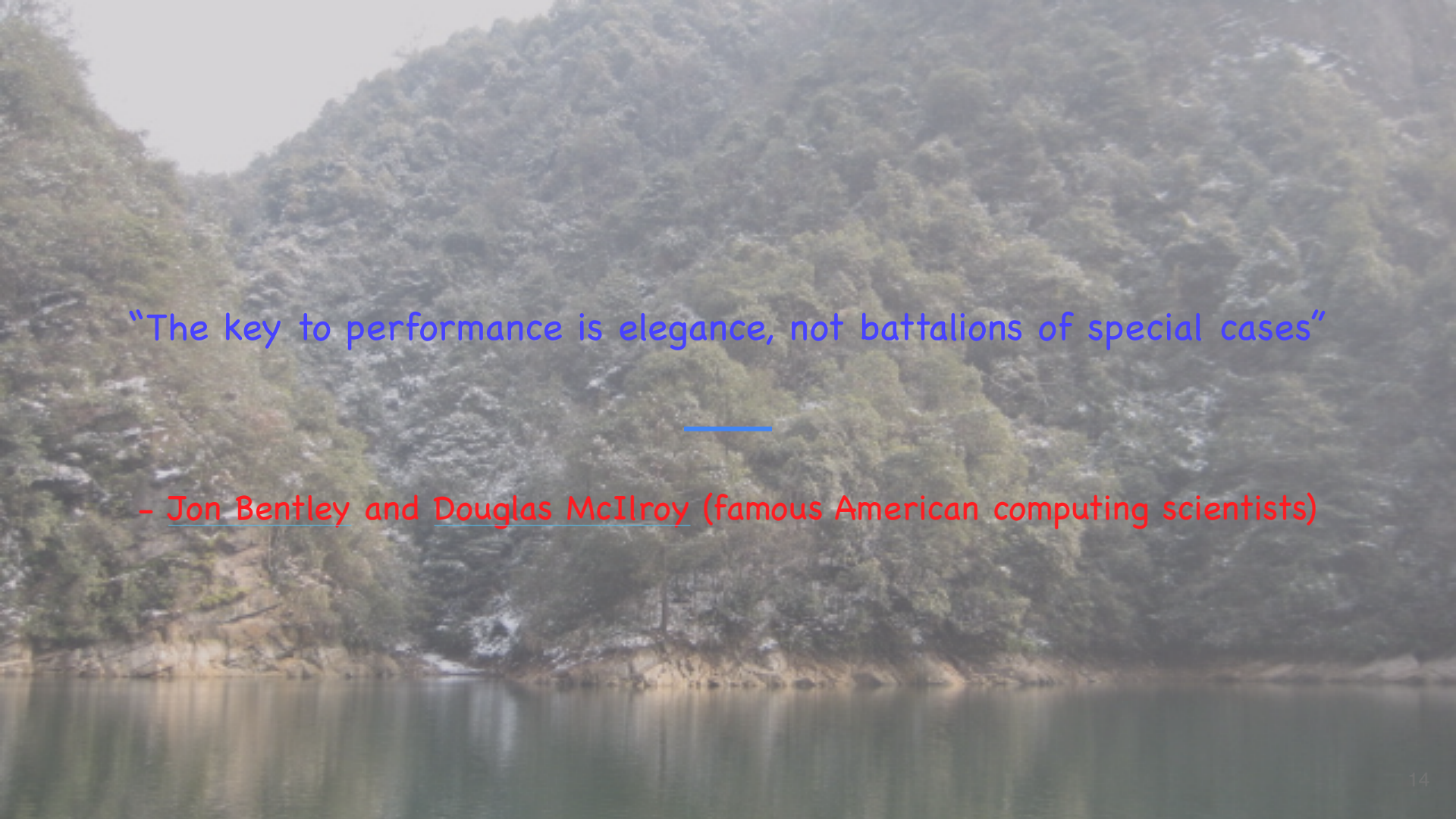
4. The simulation results and the performance parameters become inputs to the performance test cases.

No such thing as "100% Perfect" but Only 100% Science and 0% Art

Performance Engineering should be agile and iterative. Unrealistic Performance Objectives/Criteria will result in an infinite loop which leads to nowhere!

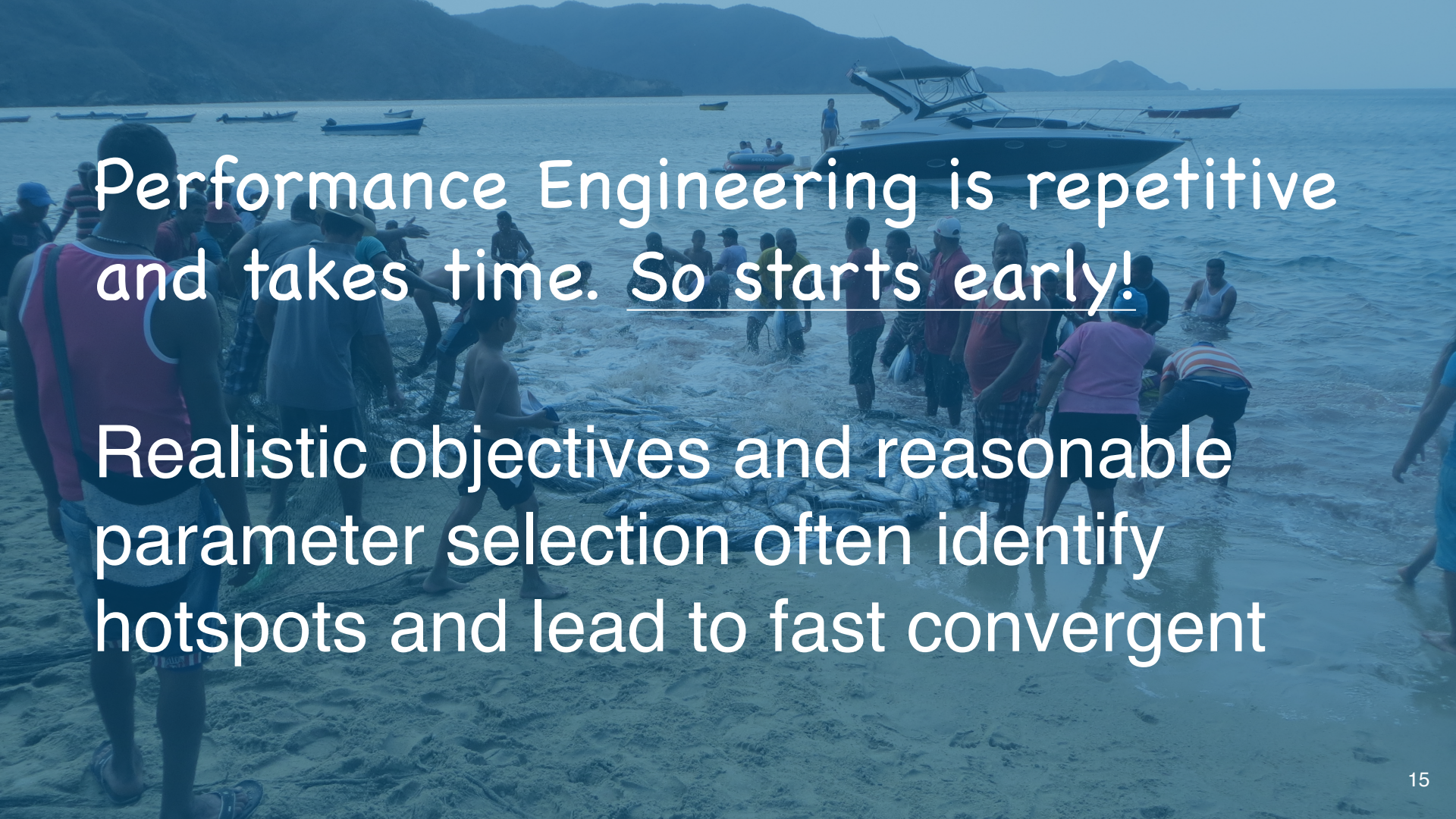
As performance parameters converge and criteria are met, your system should be "close to optimally" tuned

Both Performance Simulation/ Solver and Performance Test cases are essential tools for Performance Engineering



"The key to performance is elegance, not battalions of special cases"

– Jon Bentley and Douglas McIlroy (famous American computing scientists)



Performance Engineering is repetitive and takes time. So starts early!

Realistic objectives and reasonable parameter selection often identify hotspots and lead to fast convergent

Thanks!

Contact us:

Ken Chan
Timeleap Inc.
Toronto, Ontario, Canada

ken@timeleap.com
http://www.timeleap.com