Look Up!
Your Future is in the Cloud

James Larus
Microsoft Research

PLDI, June 2013
What is the Cloud?

Data centers at scale ↔ networked elastic computation ↔ big data
Paradigm Shift

Single computer to clusters + mobile
“Batch” and “Interactive” Cloud Computing


Cloud Computing Buzzwords

"IaaS"  Infrastructure-as-a-Service
host

"PaaS"  Platform-as-a-Service
build

"SaaS"  Software-as-a-Service
consume

http://www.silverlighthack.com/
Driving Factors

- Mobile computing
- Economies of scale
- Elastic computing
- End of Moore’s Law
- Too much data
Globally, mobile data traffic will grow 13-fold from 2012 to 2017, a compound annual growth rate of 66%.

Globally, mobile data traffic will reach 11,156,995 Terabytes (11.16 Exabytes) per month in 2017, the equivalent of 2,789 million DVDs each month or 30,746 million text messages each second.

Global mobile data traffic will grow 2.8 times faster than Global fixed IP traffic from 2012 to 2017.

Globally, mobile data traffic in 2017 will be equivalent to 771x the volume of Global mobile traffic ten years earlier (in 2007).

Globally, mobile traffic per user will reach 2,037 megabytes per month in 2017, up from 201 megabytes per month in 2012, a CAGR of 56%.
Economies of Scale of Large Datacenters

2006 comparison of very large service with mid-size: (~1000 servers):

- Large Service: $4.6/GB/year (2x in 2 Datacenters)
- Medium: $26.00/GB/year* (5.7x)
- Large Service [$13/Mb/s/mth]: $0.04/GB
- Medium [$95/Mb/s/mth]: $0.30/GB (7.1x)
- Large Service: Over 1,000 servers/admin
- Enterprise: ~140 servers/admin (7.1x)

Source: Microsoft.
Moore’s “Law” and Limiting Exponentials ...
New Bytes of Information in 2010

Source: IDC, as reported in The Economist, Feb 25, 2010

$1.2 \times 10^{21}$
Economics of Storage

Disk Storage (per gigabyte)

2000: $4.04.0576

Web Storage (per gigabyte)

2000: $10.2150

... free storage is like free puppies ...
In 2000 the Sloan Digital Sky Survey collected more data in its 1st week than was collected in the entire history of Astronomy.

By 2016 the New Large Synoptic Survey Telescope in Chile will acquire 140 terabytes in 5 days - more than Sloan acquired in 10 years.

The Large Hadron Collider at CERN generates 40 terabytes of data every second.
Genetics Gets Personal

$45,000 per Genome

$3 billion per Genome

$100-$10,000 per Genome

$500-$10,000 per Genome

3e09 bytes per person x 6e09 people = 1.8e19 bytes = 10K petabytes

(Why stop at 1 cell per person? Why stop at humans? Why stop at animals?)

Source: George Church, Harvard Medical School, as reported in IEEE Spectrum, Feb ‘10. Figures represented in USD.
Move Computation to Data(center)

Never underestimate the bandwidth of FedEx on ful disk pes hurtling down the highway. Truck
— Jim Gray, 2003

5000 2TB drives = 10 petabytes (10e16)
Next day delivery ≈ 10e6 seconds
1e10 bytes/sec ≈ 10 GB/sec ≈ 100 Gb/sec
New World, Still Has Programming Problems

Familiar problems

- Pervasive parallelism
- Partial failure
- High, variable communication latency
- Replication for reliability and throughput
- Deadlines and approximate computation

New challenges

- Multitenancy
- Services
  - Involuntary upgrades
  - Reliability, availability
  - Impact = people * rate
Pervasive Parallelism
“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable” – Leslie Lamport
High and Variable Communication Latency

Time Series (binned by 100 milliseconds)

# packets received

Core Link Utilization

CDF

EDU1
EDU3
PRV1
PRV2
CLD1
CLD2
CLD3
CLD4
CLD5
Replication for Reliability and Throughput

Eric Brewer’s CAP Theorem:
Consistency
Availability
Partition Tolerance
Choose 2
Deadlines and Approximate Computation

Quality vs. Normalized Processing Time

Bing Search Quality

He, Y., et al. Zeta: Scheduling Interactive Services with Partial Execution. SOCC, 10/12.
Multitenancy – Shared Service

Single Tenant

Multi Tenant

Tenant

Tenant

Application

Application

App Platform

App Platform

DB

DB

OS

OS

Tenant

Tenant

Application

Application

App Platform

App Platform

DB

DB

OS

OS

Tenant

Tenant

Application

Application

App Platform

App Platform

DB

DB

OS

OS
Services Must Be Reliable and Available
## What Can Programming Languages Do?

### First-class language support
- Shared memory and message passing
- Replicated data
- Failure handling
- Domain-specific languages

### Massively parallel development tools
- Failure detection and notification at scale
- Correctness and performance debugging for services

### Services life cycle support
- Strong versioning
- Live update
- Introspective monitoring and control
Existing Languages

Many languages in use, no focus on Cloud
Programming Software-Defined Networks

High-level language for programming OpenFlow networks

Paper: Wed, 4pm!
Orleans

**Framework for productive cloud software development**
- Experimental .NET library from Microsoft Research
- Runs on desktops, servers, Microsoft Azure
- Used by Microsoft for several services

**Radically simplified, prescriptive cloud programming model**
- Actors
- Asynchronous messaging
- Lightweight transactions
- Persistence
- Adaptive performance management

**Burden of correctness and performance on Orleans (not dev)**
- Visual Basic of cloud programming
Orleans Features

- Grains
- Grain activations
- Messages
- Promises
- Transactions
- Adaptive performance
- Persistence
Actor Based

Customer Grain

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>“John Doe”</td>
</tr>
<tr>
<td>Email</td>
<td>“<a href="mailto:john.doe@hotmail.com">john.doe@hotmail.com</a>”</td>
</tr>
<tr>
<td>Address</td>
<td>“123 Main St., Anywhere UR 01234”</td>
</tr>
<tr>
<td>Products</td>
<td></td>
</tr>
</tbody>
</table>

Methods

- Checkout
- AddProduct
- RemoveProduct

State

- Grain ID
- Message Queue
- Messages

Grain (actor)
Why Actors?

**Fine-grain distributed objects**
- Widely used, natural abstraction: computation as reusable service
- Isolation and message passing mirroring physical hardware

**Secure and isolated computation with clear communications**
- Singularity OS
- Computation replication

**Encapsulated and partitioned data**
- Scalability and replication

**Natural integration with persistent storage**
- Grain resides on disk until activated
In-Grain Programming Model

Field | Value
--- | ---
Name | “John Doe”
Email | “john.doe@hotmail.com”
Address | “123 Main St., Anywhere UR 01234”

Checkout
AddProduct
RemoveProduct

Grain ID
Grain ID1
Grain ID2
Radically Simple Parallel Programming

Minimize shared-memory parallelism
Parallel processes, not threads
(Can thread single task when valuable)

Explicit asynchrony
All communications is asynchronous
Error-propagation and handling are first-class constructs

Simplify challenging aspects of distributed systems
Force developers to use restricted, less error-prone parallelism
Discourage synchronous constructs that lead to poor scalability and performance

Enable automatic grain replication
Transparent performance improvement
Promises - Resolve Impedance Mismatch

Call/return is synchronous, message is asynchronous

Bind remote computation to processing of eventual result
  Communication operation returns a promise
  Caller binds promise to closure
    Evaluated when result becomes available
    Produces promise for result of closure, and so on...

Explicit representation of concurrency
  Compose stages via dataflow dependencies

Errors propagated
  Error handling can be added where needed – ’asynchronous try/catch’

(Replaced by .NET 4.5 async construct)
Replication

Scalability comes from partitioning and replication
  Grains encourage service model

Concurrent grain activations handle independent requests
  Orleans runtime dynamically replicates grains to handle load

Inconsistent can occur in shared, persistent state
  Transactions and state merging can approximate consistency at reasonable cost
State Merging

Default: “last writer wins”
Eventual consistency

Library of commutative replicated data structures
(Marc Shapiro, INRIA)
Set, tree, hashtable
Consistent semantics across distributed changes

Revisions and isolation types
(Burckhardt, Leijen, MSR)
Branch at activation
Merge activations’ updates at persistent store
Propagate updates to other activations

Transactions

ACID

Isolation

Atomicity

Consistency
Limited Transactions

**Isolation**: grain activations are isolated from activations responding to other requests

Appears as if system is processing only one request

**Atomicity**: computation either completes successfully, or no persistent state changes

If computation fails, can re-execute original request

**Consistency**: only one activation of grain allowed in transaction

Multiple messages see consistent view of grain’s state

**Not serializable!**

Local properties (atomic writes), but no global guarantees across transactions
Challenges

Distributed transactions are notoriously difficult and expensive

Use simple, non-serializable transactions

Avoid single point of conflict

Experimental: still needs refinement
Adaptive Performance Management

Grain Placement

Grain Migration
Automatic Performance Tuning

Measure performance of grain
  # of requests, latency, throughput

Create more activations if grain is heavily loaded

Shift load from overloaded servers by moving activations

Transparent to application
  Possible because of location-independent grain programing model
Experience

Intended for “Mort”, used by “Einstein”

Don’t know if it is Cloud VB

Developers like simplicity and predictability

Easy to get started
Hides complexity of underlying systems
Focus on architecture and application, not system details
Predictable performance behavior
Conclusion

Computing eras: mainframe, mini, pc, mobile, cloud

Punctuated equilibrium (Stephen Jay Gould)

Disruptive change \iff research opportunity

Language & compiler community

Many programming challenges
Strong need for better languages and tools
Do not miss the cloud opportunity

"The first bird hatched from a reptilian egg."
Backup Slides
Appropriate Size of a Grain

Reducing grain overhead enables finer-grain objects

Share resources (communication channels and OS threads) among grains in silo (process)

**Chirper**
- Simplified Twitter-like system
- ~200 lines of application code

**Horton**
- Distributed graph database

**Sparse Linear Algebra**
- Eigenvectors of large matrices (1B x 1B)
- PageRank calculation

**User Message Address book**

**Messaging intensive**

**Data intensive**

**Graph partition**

**Processor**
Persistence