Clouds Burst Horizon

J.Crowcroft/S.Hand, University of Cambridge

Cloud Computing?

- **Q**: What is Cloud Computing, exactly?
- **A**: It depends… but a rough definition might be ***on-demand internet-based computing***
  - i.e. a bunch of computers which people can use when they want, accessed via a network
- May sound a little familiar…
  - distributed [operating] systems (early 80s), cluster computing (late 80s), grid computing (90s), …
- "Cloud computing" is the Next Big Thing

Cloud Computing!

Canada
Cloud Computing

Gartner
Billion in

North Ameri

Cloud Compu
Market (Merr
Merrill Lynch ar
believes the add
computing is $1
twice Microsoft'
broken down be
Billion), platforr
Microsoft is one
important? "Azu
customer exper
Microsoft can ga
At this early sta

"After many

Cloud Computing and the Canadian Environment

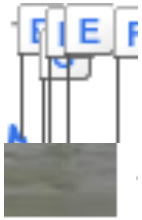
The Canadian Government's CTO of Public Works Government Services, Jirka Danek, recently presented a paper on Cloud Computing and the Canadian Environment. Essentially, the paper outlines the Canadian Government's intention to persue cloud computing. It discusses the advantages of their vast landscape and cold climate(among other things) as prime reasons for the construction of large energy efficient Cloud Computing data centers.

The following are excerpts from the report:

- The move towards Cloud Computing is inevitable and it is happening across the globe and Canada has a definite advantage on other countries around the world.

  Due to it's geographical characteristics, cooler temperatures and low-density population (particularly as one moves farther north in Canada), IT expertise, quality construction standards, legislative framework (including the Privacy Act and the Personal Information Protection and Electronic Document Act) and low-cost green energy, Canada is considered a prime location for Cloud Computing.

  Canadians Can benefit through prompt, coordinated and sustained action within Canada, across



om,
is
hise.

So what's new?

- Two key differences from previous
  - **Scale**: targeting global customer base
  - **Money**: charging (explicitly or implicitly) built in
- Three variant technologies in play:
  - Infrastructure as a Service (**IaaS**)
  - Platform as a Service (**PaaS**)
  - Software as a Service (**SaaS**)
- IaaS & PaaS explicitly charge for resources
- SaaS either bundled rental, or "free"

Not *just* hype

- Some **real customer benefits**:
  - Reduced CapEx (don't buy kit)
  - Reduced OpEx (electricity, cooling, admins)
  - Higher availability
  - Higher access bandwidth (anti-DDOS)
  - Increased flexibility ("scale out")
- **Ease of use** a key benefit for SaaS
- **Incentives for providers** due to stat mux
  - And co-locating data centers with cheap energy

What could be the problem?

1. Trust in the Cloud
   - Do components/guests trust each other
2. Trust on the Cloud
   - Why do we trust provider with our data/comp?
3. Trust by the Cloud
   - How does the cloud know who we are
4. Trust of the Cloud
   - The Cloud is a Very Big Botnet/Gun

1&2 Security, Privacy & Trust

- Cloud computing fundamentally involves using someone else's computer to run your program
- This is a **massive barrier** to many use cases
  - What if cloud platform is hacked?
  - Or can leak info to co-hosted foreign software?
  - Even worse, what if cloud provider is malicious?
- Essentially two distinct problems here:
  1. Can I trust remote system to DTRT?
  2. If not, can I still use cloud computing?

Trusting the Cloud Platform

- Basically assume cloud provider is non malicious:
  - And uses best of breed technology to provide secure isolation between multiple clients (e.g. Xen ;-)
- **Threat model**: cloud platform gets hacked
- So generally want to check:
  - *Are we talking to the right guy?* (SSL cert, TPM attest)
  - *Is he running in an 'ok' configuration?* – e.g. IBM's IMA
- Even with best solutions, vulnerabilities remain
  - Lack of spatial + temporal coverage
  - Data which "escapes" the running system
  - (sealed keys & encrypted disks some help for last)

What if we're even more paranoid?

- Assume cloud provider is (potentially) malicious
  - i.e. wants to steal our data or algorithms!!
- One possibility is *homomorphic encryption*
  - upload encrypted data, operate on encrypted data, download results, and then decrypt the answers
  - Needs $D_K(f(E_K(\langle data \rangle))) = f(\langle data \rangle)$…,where f(.) is anything
  - Recent (Gentry 2009) secure solution for unbounded comp
  - Unfortunately not really practical
    - encryption/decryption costs likely add a factor of a trillion ;-)
  - And doesn't hide algorithm (if that's the sensitive thing)
- Right now seems obfuscation is the best we can do
  - Remove data labels; scale via constant; split across nodes

Programming Frameworks

- A key aspect of cloud computing is *scale out*
  - i.e. can increase [or decrease] #compute nodes on demand
  - "elastic" response to demand can lead to lower running costs
- Support for scale out depends on underlying abstraction
- For PaaS systems, can add transparent scaling to framework
  - e.g. Azure includes "Web Roles" and "Worker Roles"
  - Instances of these run within a light weight windows environ
  - Former useful for web scaling , latter more general purpose
- For IaaS schemes (EC2, Rackspace), the default unit is a VM
  - Great for legacy/ease of use, but no built-in scale out support
  - Client-parallel systems (such as web servers/services) easy…
  - But generally need auto parallelization + distribution + FT :^)

Traditional Scale Out

- Traditional cluster solution is **message passing** (e.g. MPI)
- Allows single application (source code, or maybe even binary) to execute on SMP, NUMA, or across a cluster
- However rather intricate and low-level
  - Partitioning of computation into parallel strands is done manually by the programmer.
  - Inserting locks or barriers or signals in the correct location is done manually by the programmer
  - Failure handling is done manually by the programmer
  - Or, more likely, not!
- In general not particularly suited for situations where #compute nodes changes dynamically during execution
  - Need something both more flexible and easier to use…

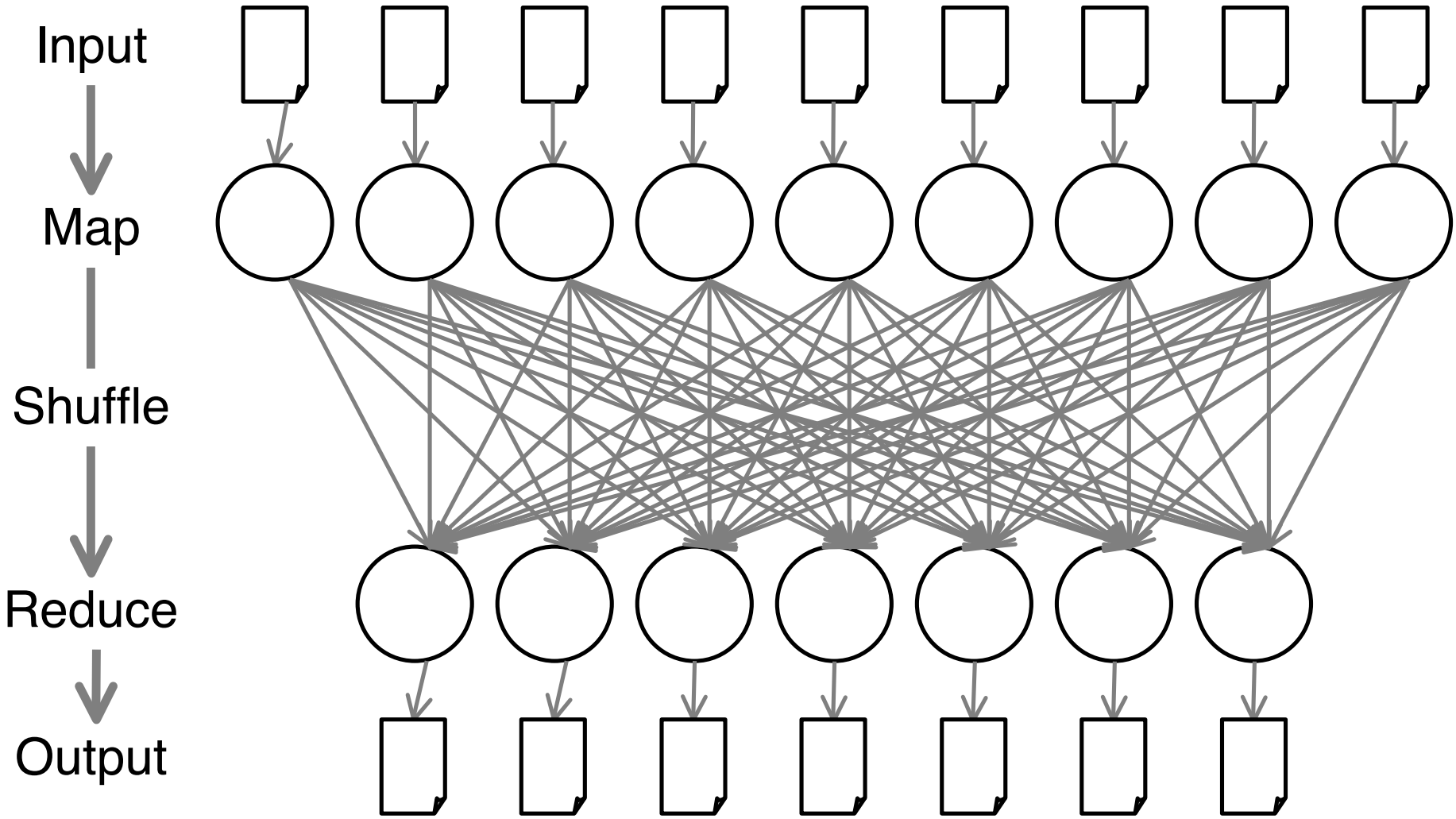
Task Parallelism

- At the basic level, consider a computation (or "job") to comprise a set of **independent** tasks
  - Job coordinator farms tasks out to workers
  - Dynamic scaling easy providing #tasks > #workers
    - If new worker arrives, give him a task
    - If a worker leaves (or fails), just re-execute the task
    - (Assumes tasks are – or can be made to be – idempotent)
  - Examples include BOINC (nee SETI@Home), Condor
- More useful if add **dependencies**…
  - i.e. allow task X to depend on tasks { Y, Z }
- … and **communication**
  - i.e. allow task A to send data to task B

Distributed Dataflow Languages

- Combining dependencies+communication leads to DDL
  - Dependencies are defined by output <-> input mappings
- A well known example is Google's **MapReduce**
  - A simple way to program data-intensive applications which (a) scale out; and (b) are fault tolerant.
  - Programmer provides just two functions
    - **map()** applied in parallel to all elements in input, produces output
    - **reduce()** applied in parallel to all elements in intermediate data
  - Inspired by functional programming, although:
    - Targeting data intensive computing
    - No implication that reduce is a traditional 'reduction'
  - In practice, use some ***m*** mappers and ***r*** reducers

MapReduce Dataflow Graph



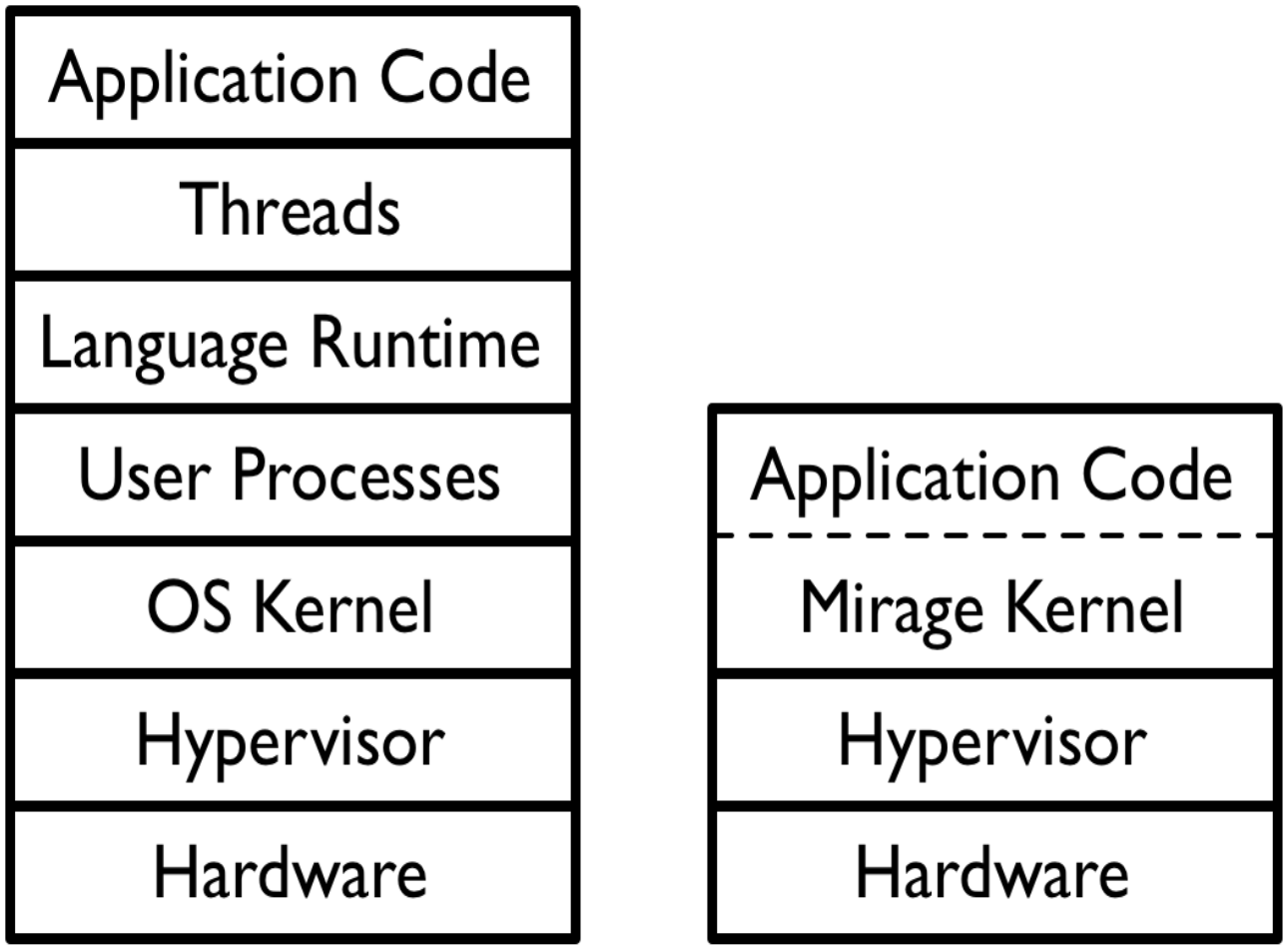
Problems with MapReduce

- MapReduce is limited in what it can express
  - precisely one dataflow pattern with params (***m***, ***r***)
  - (use of combiners allows one more pattern)
- Microsoft's Dryad extends this by allowing the dataflow to be an arbitrary finite DAG
  - However still statically determined *a priori*
  - So no **iteration**, **recursion** or **non-determinism**
- Some recent work attempts to fix this
  - E.g. SkyWriting (HotCloud'10) is a Turing complete coordination language for generating dataflow graphs
  - WIP, but some promising results – talk to me later ;-)

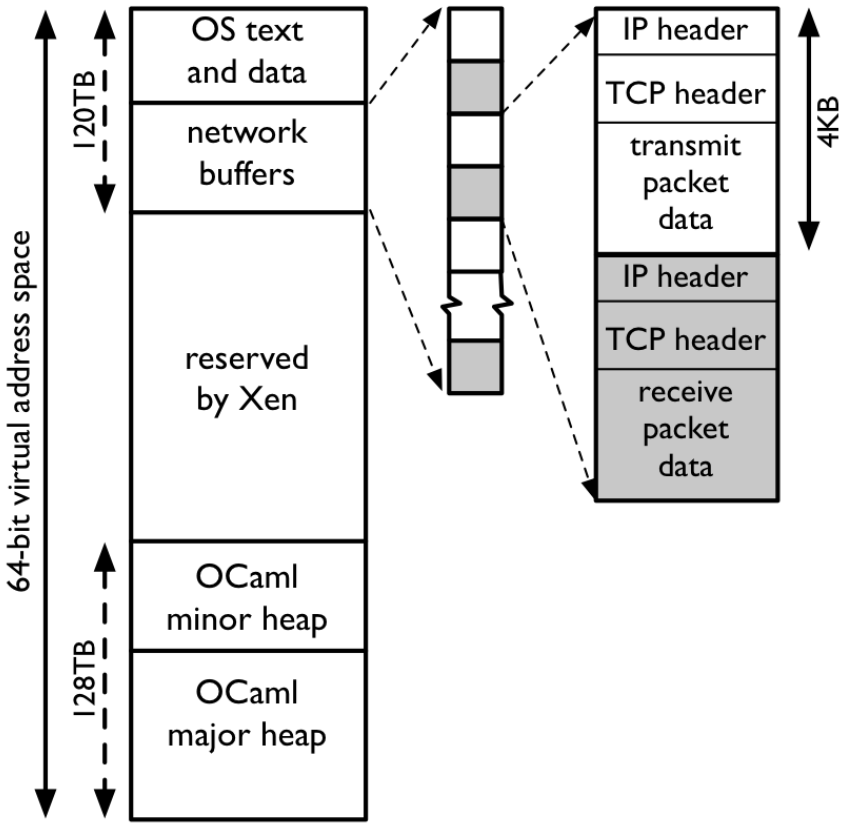
Cloud Run-time Environments

- If we move to new programming paradigms, great potential for scalability and fault tolerance
- But MapReduce/Dryad are user-space frameworks in a traditional OS (in a VM!)
  - Do we really need all these layers?
- One possibility is to build a "custom" OS for the cloud (or at least for data intensive computing)
  - E.g. Xen powers most cloud computing platforms
  - It forms a stable virtual hardware interface
  - Therefore, can compile apps directly to Xen "kernels"

MirageOS: Specialized Kernels



MirageOS: Current Design



Memory Layout

64-bit para-virtual memory layout
No context switching
Zero-copy I/O to Xen
Super page mappings for heap

Concurrency

Cooperative threading and events
Fast inter-domain communication
Works across **cores and hosts**

Future Directions

- MirageOS is just one attempt
  - (are VMMs uKerns done right? ☺
- More generally, need to consider multi-scale
  - Computing across cloud, mobiles & desktops
  - Can easily extend to multi core / many core too
- Challenges remain with programmability, interactivity, and debugging …

Data Management & Movement

- How to transfer large amounts of data from client to cloud?
- State of the art quite basic
  - reference images + scp/REST… or FedEx!
- One possibility is **global-scale de-duplication**
  - Divide data into chunks, hash to get code point
  - Cloud providers store chunks indexed by hash value
  - (Basically just rsync / LBFS… but at a much larger scale)
- Has many advantages:
  - Can vastly reduce client->cloud transfer time (and vice versa)
  - Easily extended to allow wide-area "storage migration"
- However some challenges in building efficient index lookup
- And some tensions when interacting with encryption...

Personal Cloud Storage

- May store personal stuff in the cloud too
  - e.g. email, docs, photos, blogs, …
- Many services today (Google, FB, Flickr, …)
  - But unclear who actually "owns" the data
  - Raises issues about privacy (data mining), durability (backup), provenance (invention), legal liability, and so on
- Can technically build private cloud storage via encryption/steganography & multiple cloud services
  - User can now handle durability, provenance + liability
  - But makes search and, generally, organization problematic
  - Plus also breaks the current business model :^)
- Open question how to attain best of both worlds

3. Trust by Cloud

- IP is anonymous –
- No accountability
  - How do you know who a client is
  - Where a client is (GeoMapping)
    - Is done as part f targetted advertising
    - Usually works for DSL access (not on BT☺)
    - Does work for cellular data access
- Sybil attacks, DDoS, Botnet etc etc etc

3++ Money Money Money

- The cloud today is a mish-mash of explicit and implicit payment, quotas and penalties
- Unclear [to me at last] what will emerge here
  - Perfect market moving toward marginal costs?
- One interesting data point are EC2 spot prices:
  - Typically 20%-50% cheaper than "on demand"…
  - But same – or higher – than "reserved"
  - So already scope for a secondary market!
- If you're a capitalist, this might be good news…
  - But as a programmer or systems designer, this is a mess
  - (can hardly manage resources + QoS traditional clusters)
- But this does help with 3 (accountability☺)

4. Trust of Cloud

- The cloud is a Very Big Gun
- Currently not pointed at anyone
- Fb, Google, Amazon all have
  - >500k machines per data center
  - Multiple 10Gbps access to multiple ISPs
- A zero day 0wn on them would provide the worlds biggest botnet
  - Easily enough to take down national infrastructures for a while

4++ What to do?

- Cloud providers interested in not being 0wned
- But may be hard for them to tell
  - While slice/resource management/SLA help limit scope of a guest/customer…
  - A sybil could have a lot of slices
  - Elastic Computing model makes this easier
  - (lull provider into false sense of security, then wham!)
- **Need backup network to get in to fix things**

And so to wrap up…

- Cloud computing is here
  - Whether we want it or not!
- Considerable hype, but also many challenges:
  - How can trust third party cloud providers?
  - How can use them when we don't trust them?
  - How can efficiently and effectively program applications to run across multiple scales, multiple platforms, and multiple failure modes?
  - How can we manage data on a global scale?
  - And how do we make the economics work!

And are we there yet?

- No – things are getting worse.
  - 4G is all IP, so is part of the target and cellular is no longer separate
  - Internet of Things hype is leading organisations to connect many SCADA systems unprotected
  - We have lots of accountable internet architectures as well as some controlled sane anonymity tech, but no-ones really deploying it.
- **I guess it will take a Very Bad Day to make people wake up and fix this all**