

Programming Models for Technical Computing on Clouds and Supercomputers (aka HPC)

May 7 2012

Cloud Futures 2012

May 7–8, 2012 , Berkeley, California, United States

Geoffrey Fox

gcf@indiana.edu

Indiana University Bloomington

Dennis Gannon

Microsoft



<https://portal.futuregrid.org>

Science Computing Environments

- **Large Scale Supercomputers** – Multicore nodes linked by high performance low latency network
 - Increasingly with GPU enhancement
 - Suitable for highly parallel simulations
- **High Throughput Systems** such as European Grid Initiative EGI or Open Science Grid OSG typically aimed at pleasingly parallel jobs
 - Can use “cycle stealing”
 - Classic example is **LHC data analysis**
- **Grids** federate resources as in EGI/OSG or enable convenient access to multiple backend systems including supercomputers
 - **Portals** make access convenient and
 - **Workflow** integrates multiple processes into a single job
- Specialized **visualization, shared memory parallelization** etc. machines



Some Observations

- **Distinguish HPC (Supercomputer) machines and HPC problems**
- **Classic HPC machines** as MPI engines offer highest possible performance on closely coupled problems
- **Clouds** offer from different points of view
 - On-demand service (**elastic**)
 - **Economies of scale from sharing**
 - Powerful new **software models** such as **MapReduce**, which have **advantages over classic HPC environments**
 - **Plenty of jobs** making it attractive for students & curricula
 - **Security** challenges
- **HPC problems** running well on clouds have above advantages
- Note 100% utilization of Supercomputers makes elasticity moot for capability (very large) jobs and makes capacity (many modest) use not be on-demand
- Need **Cloud-HPC Interoperability**



Clouds and Grids/HPC

- Synchronization/communication Performance
Grids > Clouds > Classic HPC Systems
- Clouds naturally execute effectively Grid workloads but are less clear for closely coupled HPC applications
- **Service Oriented Architectures** and **workflow** appear to work similarly in both grids and clouds
- May be for immediate future, science supported by a mixture of
 - **Clouds** – some practical differences between private and public clouds – size and software
 - **High Throughput Systems** (moving to clouds as convenient)
 - **Grids** for distributed data and access
 - **Supercomputers** (“MPI Engines”) going to exascale



What Applications work in Clouds

- **Pleasingly parallel** applications of all sorts analyzing roughly independent data or spawning independent simulations
 - **Long tail** of science
 - Integration of distributed sensors (**Internet of Things**)
- **Science Gateways** and portals
- **Workflow** federating clouds and classic HPC
- **Commercial and Science Data analytics** that can use MapReduce (some of such apps) or its **iterative** variants (most other data analytics apps)
- Which applications are using clouds?
 - Many demonstrations – see today, Venus-C, OOI, HEP
 - 50% of applications on FutureGrid are from Life Science but
 - There is more computer science than total applications on FutureGrid
 - Locally Lilly corporation is major commercial cloud user (for drug discovery) but Biology department is not



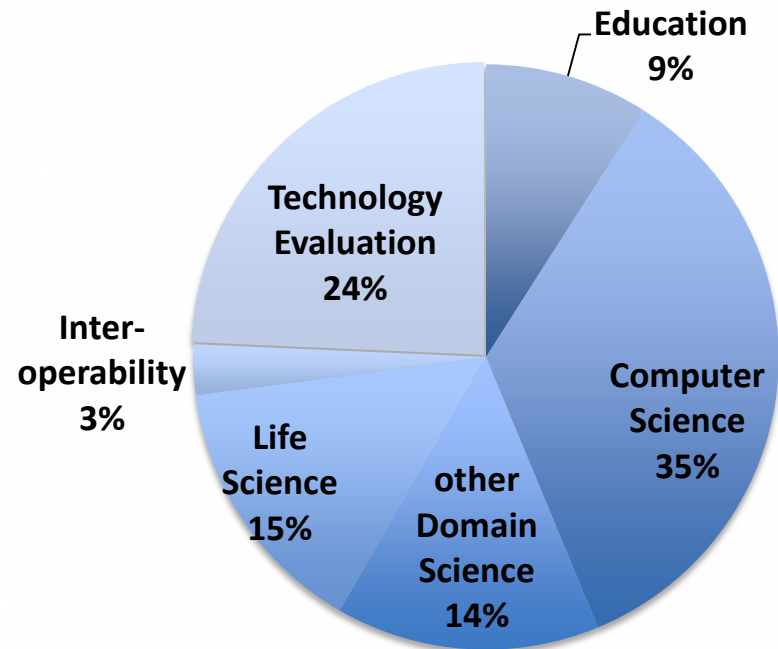
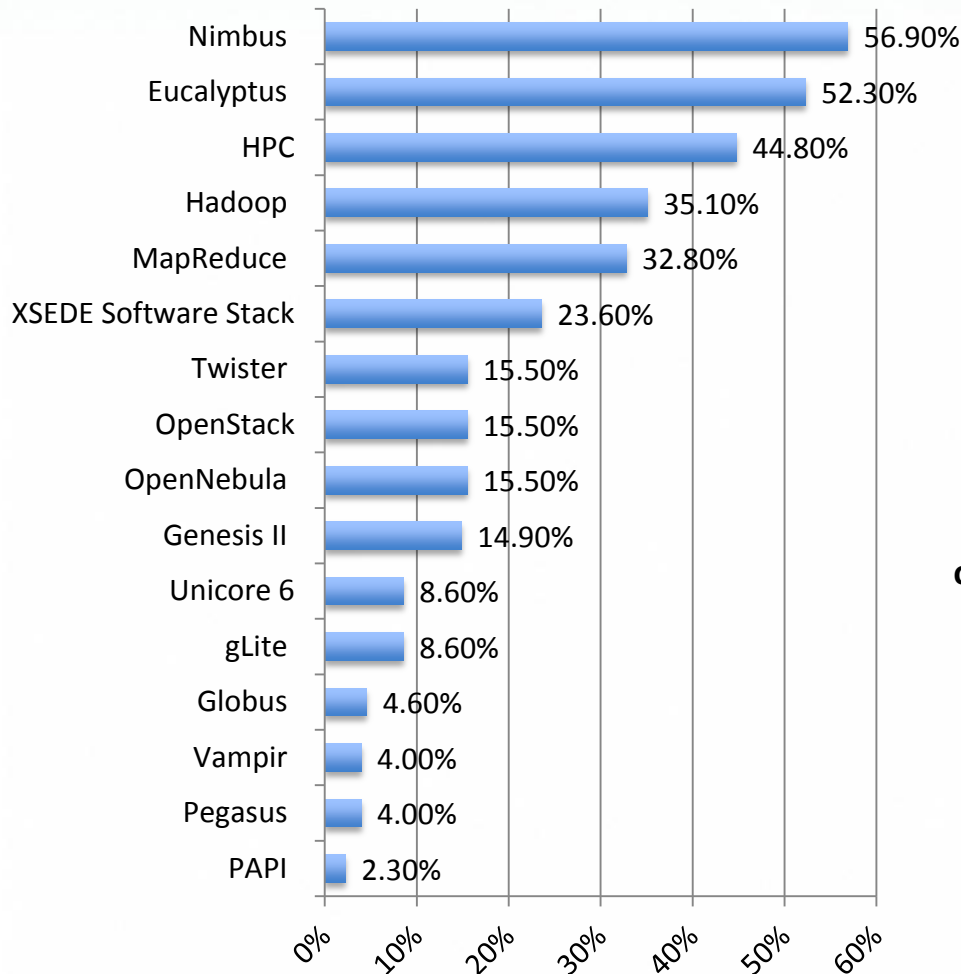
What is FutureGrid?

- The **FutureGrid** project mission is to **enable experimental work** that advances:
 - a) Innovation and scientific understanding of **distributed computing and parallel computing paradigms**,
 - b) The **engineering science of middleware** that enables these paradigms,
 - c) The use and drivers of these paradigms by **important applications**, and,
 - d) The **education** of a new generation of students and workforce on the use of these paradigms and their applications.
- The **implementation** of mission includes
 - **Distributed flexible hardware** with supported use
 - Identified **IaaS and PaaS** “core” software with supported use
 - **Outreach**
- **~4500 cores** in 5 major sites



Distribution of FutureGrid Technologies and Areas

- 200 Projects



Project Ref ID	Project Lead	Title	Institution	Date Started	Keywords	Recent FutureGrid Projects
[FG-P213]	Massimo Canonico	Cloud Computing class - second edition	University of Piemonte Orientale, Computer Science Department	04/28/2012	teaching, educational	
[FG-P212]	Epaphras Matsangaise	Cloud Computing : Confidentiality and Integrity of data	Indiana University, School Of Informatics	04/28/2012	Cloud Computing	
[FG-P211]	Zhan Wang	Performance evaluation of cloud storage placement	George Mason University, Center for Secure Information Systems	04/25/2012	cloud storage, placement	
[FG-P210]	Adetunji Anthony Adeleke	Exploring and Cataloging Cloud Computing Security issues via FutureGrid	IUPUI, Informatics	04/25/2012	security, cloud, computing, taxonomy, futuregrid, catalogue	
[FG-P209]	Karolina Sarnowska-Upton	Quantifying User Effort to Migrate MPI Applications	University of Virginia, Computer Science	04/25/2012	MPI, usability testing, migration	
[FG-P208]	Junzhou Wang	genomic_mapping	Indiana University Bloomington, Biology/MCDB/plant science/Pikaard's Lab	04/25/2012	genomic,mapping	
[FG-P207]	David Young	Disaster Recovery and Management using Twitter and MapReduce	IUPUI, Computer Science Department	04/07/2012	MapReduce,Twitter,Distributed Computing	
[FG-P206]	Seung-Hee Bae	Parallel Performance of MDS Dimension Reduction Method	Indiana University, Pervasive Technology Institute	04/05/2012	Parallel Performance, MDS, Dimension Reduction	
[FG-P205]	Ryan Newton	Parallelization of heterogeneous workloads for Imaging Genomic Browser	Indiana University, School of Informatics and Computer Science	04/05/2012	Brain Imaging, Genome Analysis, Languages and Compilers, Haskell	
[FG-P204]	Richard Knepper	GFFS Testing for XSEDE pilot	Indiana University, Research Technologies Campus Bridging	04/03/2012	GFFS, Genesis, pilot	
[FG-P203]	adnan ozsoy	Compression on GPUs	Indiana University Bloomington, School of Informatics	04/02/2012	compression, lzss, cuda, gpu	
[FG-P202]	Ifeanyi Egwuotuoha	Fault Tolerance of HPC systems	University of Sydney, Electrical and Information Engineering	04/02/2012	HPC, cloud, fault tolerance	
[FG-P197]	Manjunath Kadaba Sathanathan	comparision of performance with respect to resource allocation between cloudsim and actual cloud	East Point College of Engineering, Computer Science Department	04/02/2012	resource management, cloudsim	
[FG-P201]	Preston Smith	EXTENCI Testing, Validation, and Performance	Purdue University, Rosen Center for Advanced Computing	03/24/2012	Virtual Machines, VM Management, VM Deployment, VM Portal	
[FG-P200]	Guangchen Ruan	MapReduce Based Ray Tracing Class Project	Indiana University at Bloomington, PTI	03/24/2012	Ray Tracing, MapReduce, Cloud Computing	
[FG-P198]	Richard Knepper	XSEDE Campus Bridging Rocks Roll testing	Indiana University, Research Technologies Campus Bridging	03/24/2012	software, rocks rolls, xsede	
[FG-P196]	Barani S	elastic site	Governmrnt College Of Technology,Coimbatore, Information Technology	03/24/2012	elastically extending resources using cloud,job scheduling,queue sensor,nimbus toolkit,resource manager	
[FG-P194]	Art Vandenberg	SGVO Cloud Options Working Group	Georgia State University, Information System & Technology	03/13/2012	SGVO, OSG, software installations, testing	
[FG-P193]	Thomas William	Efficient deadlock-free routing	GWT-TUD GmbH, IT	03/03/2012	InfiniBand, single-source-shortest-path, deadlock, routing, MPI	
[FG-P192]	Jay Larson	Climate Data Analytics Using MapReduce	Argonne National Laboratory, Mathematics and Computer Science Division	03/03/2012	Climate, MapReduce, Data Analysis, Data Mining	
[FG-P191]	Prof. Jay Wang	UCF EEL6938 Data-intensive computing	University of Central Florida, Department of Electrical	03/01/2012	MapReduce, High performance computing, Analytics,	

Parallelism over Users and Usages

- “**Long tail of science**” can be an important usage mode of clouds.
- In some areas like particle physics and astronomy, i.e. “**big science**”, there are just a few major instruments generating now petascale data driving discovery in a coordinated fashion.
- In other areas such as genomics and environmental science, there are many “**individual**” **researchers** with distributed collection and analysis of data whose total data and processing needs can match the size of big science.
- **Clouds** can provide scaling convenient resources for this important aspect of science.
- Can be **map only** use of MapReduce if different usages naturally linked e.g. exploring docking of multiple chemicals or alignment of multiple DNA sequences
 - Collecting together or summarizing multiple “maps” is a **simple Reduction**

Internet of Things and the Cloud

- It is projected that there will soon be 50 billion devices on the Internet. Most will be small sensors that send streams of information into the cloud where it will be processed and integrated with other streams and turned into knowledge that will help our lives in a million small and big ways.
- It is not unreasonable for us to believe that we will each have our own cloud-based personal agent that monitors all of the data about our life and anticipates our needs 24x7.
- The cloud will become increasingly important as a controller of and resource provider for the Internet of Things.
- As well as today's use for smart phone and gaming console support, "smart homes" and "ubiquitous cities" build on this vision and we could expect a growth in cloud supported/controlled robotics.
- Natural parallelism over "things"



Internet of Things: Sensor Grids

A pleasingly parallel example on Clouds

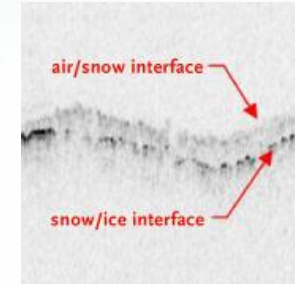
- 🌐 A **sensor** (“**Thing**”) is any source or sink of time series
 - 🌐 In the thin client era, smart phones, Kindles, tablets, Kinects, web-cams are sensors
 - 🌐 Robots, distributed instruments such as environmental measures are sensors
 - 🌐 Web pages, Googledocs, Office 365, WebEx are sensors
 - 🌐 Ubiquitous Cities/Homes are full of sensors
 - 🌐 They have IP address on Internet
- 🌐 Sensors – being intrinsically distributed are **Grids**
- 🌐 However natural implementation uses **clouds** to consolidate and control and collaborate with sensors
- 🌐 Sensors are typically “small” and have **pleasingly parallel cloud implementations**



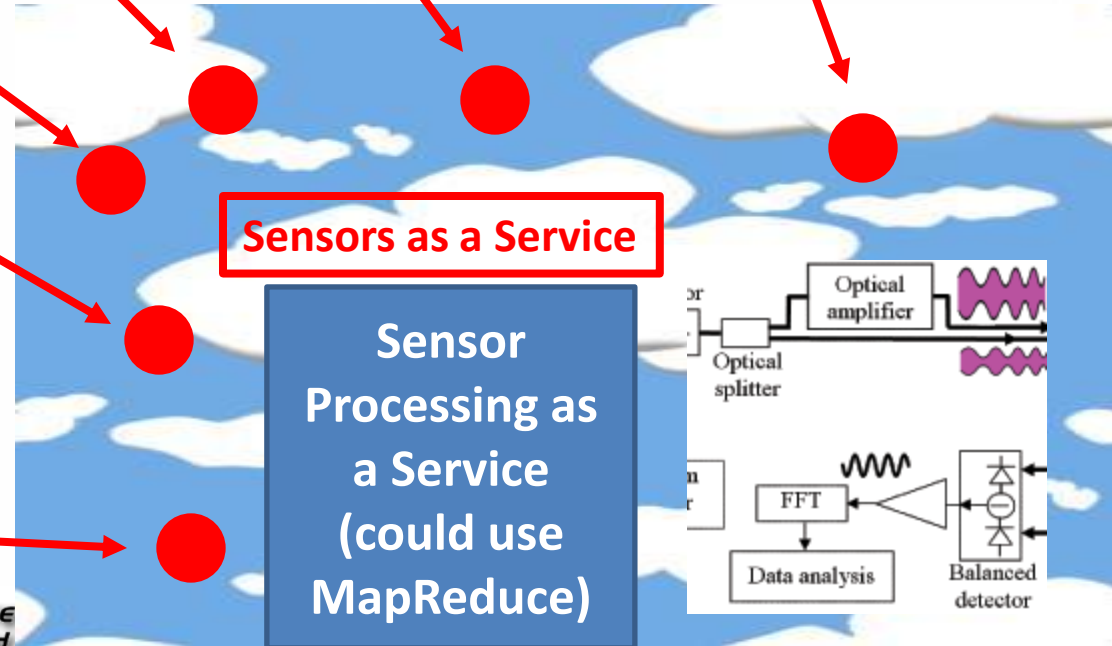
Sensors as a Service



Output Sensor



A larger sensor



Portal/Gateway

- “Just a web role” supporting back end services
- Often used to support multiple users accessing a relatively modest size computation
- So cloud suitable implementation

Workflow

- Loosely coupled orchestrated links of services
- Works well on Grids and Clouds as coarse grain (a few large messages between largish tasks) and no tight synchronization

Classic Parallel Computing

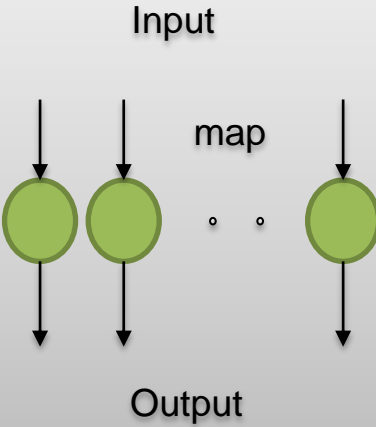
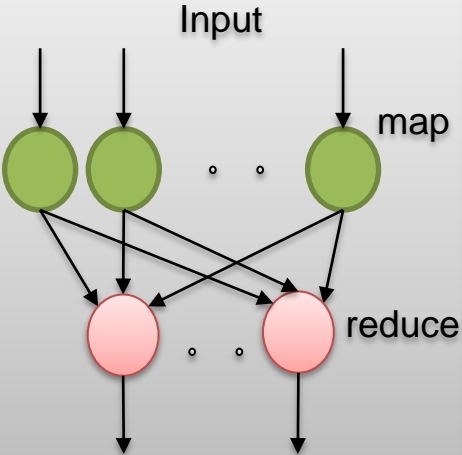
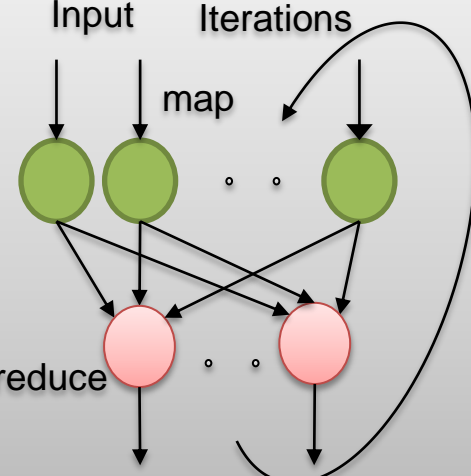
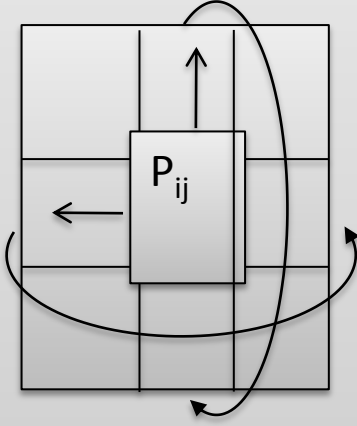
- **HPC:** Typically SPMD (Single Program Multiple Data) “maps” typically processing particles or mesh points interspersed with multitude of low latency messages supported by specialized networks such as Infiniband
 - Often run large capability jobs with 100K cores on same job
 - National DoE/NSF/NASA facilities run 100% utilization
 - Fault fragile and cannot tolerate “outlier maps” taking longer than others
- **Clouds:** MapReduce has asynchronous maps typically processing data points with results saved to disk. Final reduce phase integrates results from different maps
 - Fault tolerant and does not require map synchronization
 - **Map only** useful special case
- **HPC+Clouds:** Iterative MapReduce caches results between “MapReduce” steps and supports SPMD parallel computing with large messages as seen in parallel linear algebra need in clustering and other data mining



Commercial “Web 2.0” Cloud Applications

- Internet search, Social networking, e-commerce, cloud storage
- These are **larger systems than used in HPC** with huge levels of parallelism coming from
 - Processing of **lots of users** or
 - An **intrinsically parallel** Tweet or Web search
- **MapReduce is suitable** (although Page Rank component of search is parallel linear algebra)
- **Data Intensive**
- Do not need microsecond messaging latency

4 Forms of MapReduce

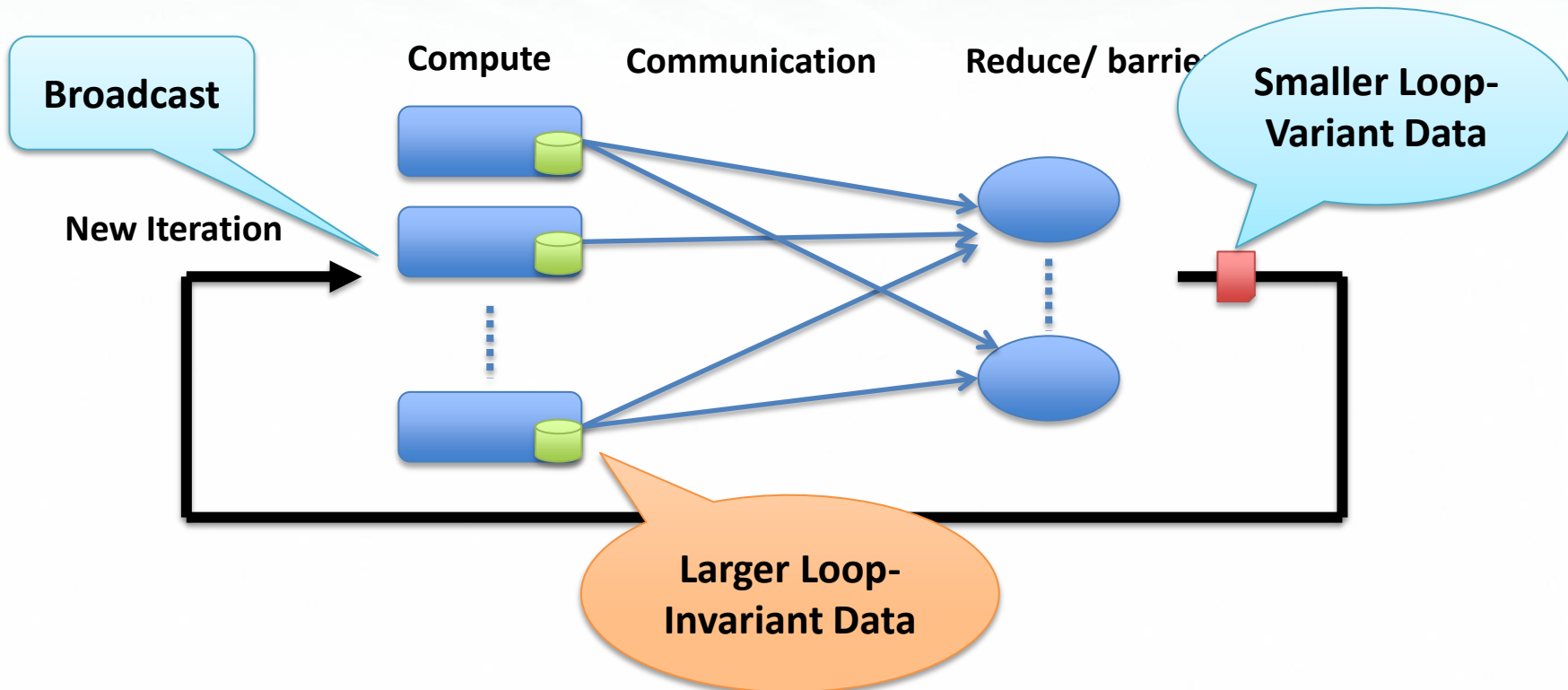
(a) Map Only	(b) Classic MapReduce	(c) Iterative MapReduce	(d) Loosely Synchronous
			
<p>BLAST Analysis Parametric sweep Pleasingly Parallel</p>	<p>High Energy Physics (HEP) Histograms Distributed search</p>	<p>Expectation maximization Clustering e.g. Kmeans Linear Algebra, Page Rank</p>	<p>Classic MPI PDE Solvers and particle dynamics</p>
<p style="text-align: center;">← Domain of MapReduce and Iterative Extensions →</p>			<p style="text-align: center;">MPI</p>

Data Intensive Iterative Applications I

- Important class of (**Data analytics**) applications
 - **Data mining, machine learning** – often with **linear algebra at core**
 - **Expectation maximization**
 - Driven by data deluge & emerging fields

```
k ← 0;
MAX ← maximum iterations
 $\delta^{[0]}$  ← initial delta value
while ( k < MAX_ITER || f( $\delta^{[k]}$ ,  $\delta^{[k-1]}$ ) )
  foreach datum in data
     $\beta$ [datum] ← process (datum,  $\delta^{[k]}$ )
  end foreach
   $\delta^{[k+1]}$  ← combine( $\beta$ [])
  k ← k+1
end while
```

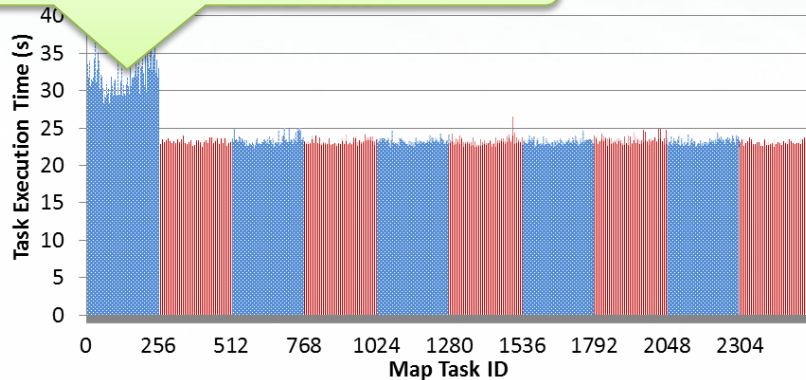
Data Intensive Iterative Applications II



- Structure from (Iterative) MapReduce point of view

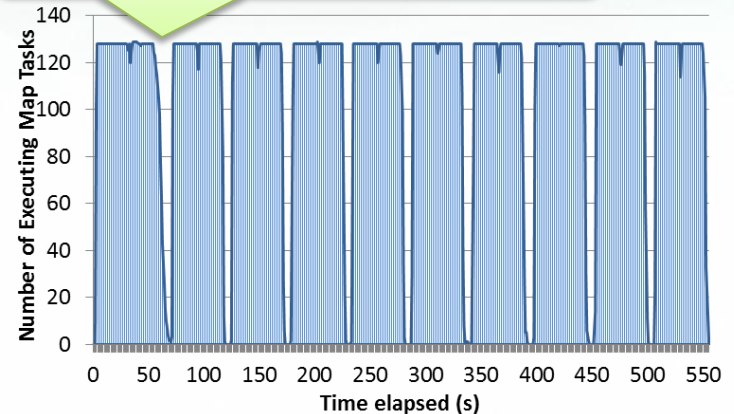
Performance – Kmeans Clustering

First iteration performs the initial data fetch

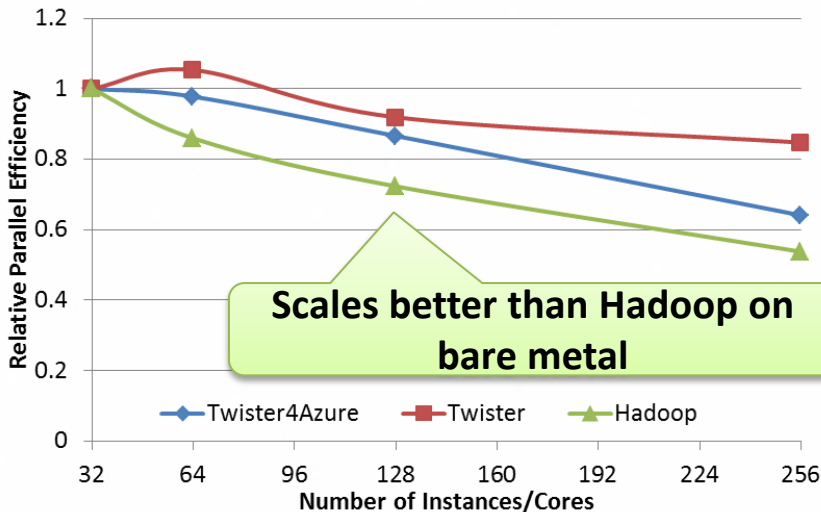


Task Execution Time Histogram

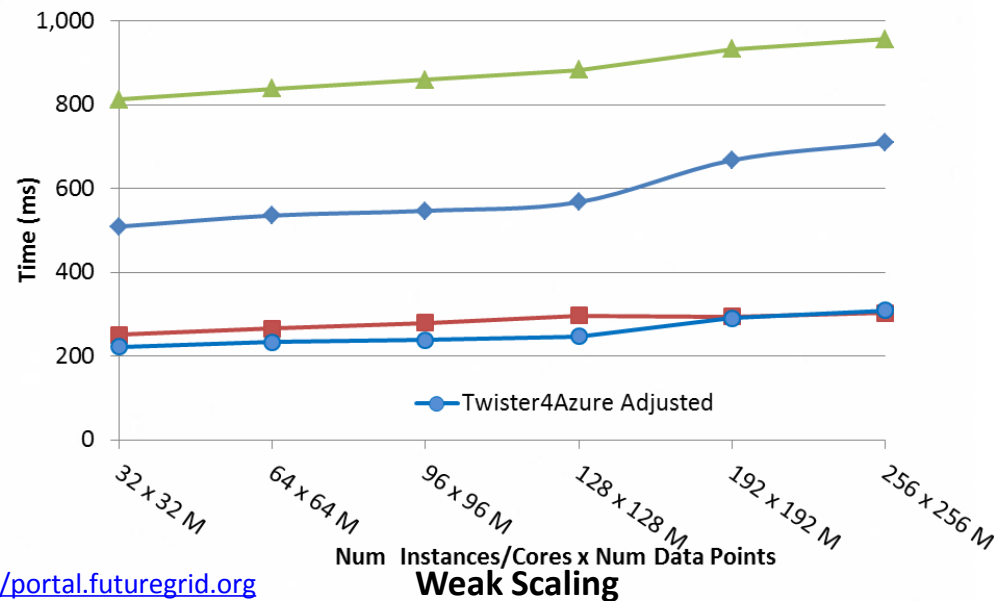
Overhead between iterations



Number of Executing Map Task Histogram



Strong Scaling with 128M Data Points



Weak Scaling



Summary: Usage modes of Clouds

- **Large Scale internally parallel**
 - Internet Search or large BLAST problem
- **Pleasingly parallel over users**
 - E-commerce or Long Tail of Science
- **Pleasing parallel over usages** (perhaps for same user)
 - Internet of Things or parameter searches
- **Iterative parallel** algorithms with large messages
 - Data mining
- **Workflow**
 - Orchestrate multiple services
- **Portals**
 - Web interface to the above modes

What to use in Clouds

- HDFS style **file system** to collocate data and computing
- **Queues** to manage multiple tasks
- **Tables** to track job information
- **MapReduce** and **Iterative MapReduce** to support parallelism
- **Services** for everything
- **Portals** as User Interface
- **Appliances** and **Roles** as customized images
- Software environments/tools like **Google App Engine**, **memcached**
- **Workflow** to link multiple services (functions)



What to use in Grids and Supercomputers?

- **Portals** and **Workflow** as in clouds
- **MPI** and **GPU/multicore threaded** parallelism
- **Services** in Grids
- **Wonderful libraries** supporting parallel linear algebra, particle evolution, partial differential equation solution
- **Parallel I/O** for high performance in an application
- **Wide area File System** (e.g. Lustre) supporting file sharing
- This is a rather different style of **PaaS** from clouds – **should we unify?**



Is PaaS a good idea?

- If you have **existing code**, PaaS may not be very relevant immediately
 - Just need **IaaS** to put code on clouds
- But surely it must be good to offer **high level tools**?
- For example, Twister4Azure (see tomorrow's talk) built on top of Azure **tables, queues, storage**
- Historically **HPCC** 1990-2000 built MPI, libraries, (parallel) compilers ..
- **Grids** 2000-2010 built federation, scheduling, portals and workflow
- **Clouds** 2010-.... have an exciting interest in powerful programming models

How to use Clouds I

- 1) Build the application as a service.** Because you are deploying one or more full virtual machines and because clouds are designed to host web services, you want your application to support multiple users or, at least, a sequence of multiple executions.
 - If you are not using the application, scale down the number of servers and scale up with demand.
 - Attempting to deploy 100 VMs to run a program that executes for 10 minutes is a waste of resources because the deployment may take more than 10 minutes.
 - To minimize start up time one needs to have services running continuously ready to process the incoming demand.
- 2) Build on existing cloud deployments.** For example use an existing MapReduce deployment such as Hadoop or existing Roles and Appliances (Images)

How to use Clouds II

- 3) **Use PaaS if possible.** For platform-as-a-service clouds like Azure use the tools that are provided such as queues, web and worker roles and blob, table and SQL storage.
- 3) Note HPC systems don't offer much in PaaS area
- 4) **Design for failure.** Applications that are services that run forever will experience failures. The cloud has mechanisms that automatically recover lost resources, but the application needs to be designed to be fault tolerant.
- In particular, environments like MapReduce (Hadoop, Daytona, Twister4Azure) will automatically recover many explicit failures and adopt scheduling strategies that recover performance "failures" from for example delayed tasks.
 - One expects an increasing number of such Platform features to be offered by clouds and users will still need to program in a fashion that allows task failures but be rewarded by environments that transparently cope with these failures. (Need to build more such robust environments)

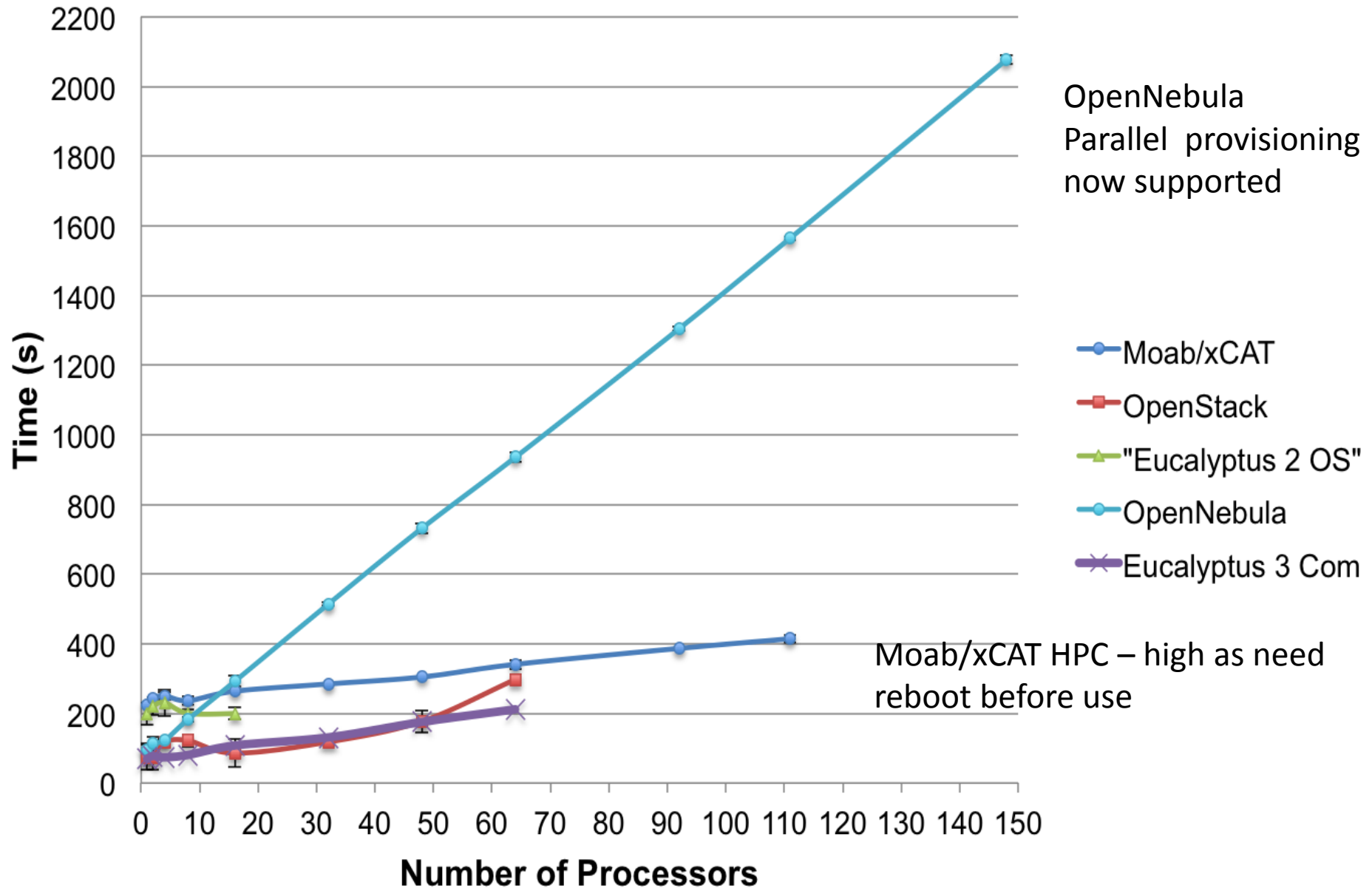
How to use Clouds III

- 5) **Use as a Service where possible.** Capabilities such as SQLaaS (database as a service or a database appliance) provide a friendlier approach than the traditional non-cloud approach exemplified by installing MySQL on the local disk.
- Suggest that many prepackaged aaS capabilities such as **Workflow as a Service** for eScience will be developed and simplify the development of sophisticated applications.
- 6) **Moving Data is a challenge.** The general rule is that one should move computation to the data, but if the only computational resource available is a the cloud, you are stuck if the data is not also there.
- Persuade Cloud Vendor to host your data free in cloud
 - Persuade Internet2 to provide good link to Cloud
 - Decide on Object Store v. HDFS style (or v. Lustre WAFS on HPC)

Private Clouds

- Define as non commercial cloud used to support science
- **What does it take to make private cloud platforms competitive with commercial systems?**
- Plenty of work at **VM management level** with Eucalyptus, Nimbus, OpenNebula, OpenStack
 - Only now maturing
 - Nimbus and OpenNebula pretty solid but not widely adopted in USA
 - OpenStack and Eucalyptus recent major improvements
- **Open source PaaS** tools like Hadoop, Hbase, Cassandra, Zookeeper but not integrated into platform
- **Need dynamic resource** management in a “not really elastic” environment as limited size
- **Federation** of distributed components (as in grids) to make a decent size system

Templated Dynamic Provisioning



Some Research Challenges – I

- Design algorithms that can **exploit/tolerate cloud features**
 - Elastic access to resources
 - Use few large messages – not lots of small ones
 - Fault tolerant
 - Use library of roles and appliances
 - Exploit platforms (queues, tables) and XaaS
- **Classify** and measure **performance** of these algorithms/applications
- **Improve performance** of clouds
- Many **security** issues
- Understand needed **standards**

Helped by Manish Parashar



Some Research(&D) Challenges – II

- Improve **MapReduce** so it
 - Offers HPC Cloud interoperability
 - Polymorphic reductions (collectives) exploiting all types of networks
 - Supports scientific data and algorithms
- Develop **storage model** to support cloud computing enhanced data repositories
- Understand **federation of multiple clouds** and support of hybrid algorithms split across clouds (e.g. for security or geographical reason)
 - Private clouds are not likely to be on huge scale of public clouds
 - **Cloud bursting** important federated system (private + public)
- Bring **commercial cloud PaaS** to HPC and academic clouds
- **Fault tolerance, high availability, energy efficiency** (green clouds)
- **Train people** for the 14 million cloud jobs expected by 2015



Architecture of Data Repositories?

- Traditionally governments set up repositories for data associated with particular missions
 - For example EOSDIS (Earth Observation), GenBank (Genomics), NSIDC (Polar science), IPAC (Infrared astronomy)
 - LHC/OSG computing grids for particle physics
- This is complicated by volume of data deluge, distributed instruments as in gene sequencers (maybe centralize?) and need for intense computing like Blast
 - i.e. **repositories need lots of computing?**

Clouds as Support for Data Repositories?

- The **data deluge** needs cost effective computing
 - Clouds are by definition cheapest
 - Need data and computing co-located
- **Shared resources** essential (to be cost effective and large)
 - Can't have every scientists downloading petabytes to personal cluster
- Need to reconcile **distributed** (initial source of) **data** with shared analysis
 - Can move data to (discipline specific) clouds
 - How do you deal with multi-disciplinary studies
- **Data repositories of future will have cheap data and elastic cloud analysis support?**
 - Hosted free if data can be used commercially?

Outreach

- Papers are Programming Paradigms for Technical Computing on Clouds and Supercomputers (Fox and Gannon)
http://grids.ucs.indiana.edu/ptliupages/publications/Cloud%20Programming%20Paradigms_for_Futures.pdf
<http://grids.ucs.indiana.edu/ptliupages/publications/Cloud%20Programming%20Paradigms.pdf>
- **Science Cloud Summer School** July 30-August 3 offered virtually
 - Aiming at computer science and application students
 - Lab sessions on commercial clouds or FutureGrid
- Would like volunteers interested in talking or attending!



Using Clouds in a Nutshell

- **High Throughput Computing**; pleasingly parallel; grid applications
- **Multiple users** (long tail of science) and **usages** (parameter searches)
- **Internet of Things** (Sensor nets) as in cloud support of smart phones
- **(Iterative) MapReduce** including “most” data analysis
- Exploiting **elasticity** and **platforms** (HDFS, Queues ..)
- Use **services**, **portals** (gateways) and **workflow**
- Good Strategies:
 - Build the application **as a service**;
 - Build on existing cloud deployments such as **Hadoop**;
 - **Use PaaS** if possible;
 - **Design for failure**;
 - **Use as a Service** (e.g. SQLaaS) where possible;
 - Address Challenge of **Moving Data**