Belle Monte-Carlo production on the Amazon EC2 cloud

Martin Sevior, Tom Fifield (University of Melbourne)
Nobuhiko Katayama (KEK)

17th International Conference on Computing in High Energy and Nuclear Physics
21 - 27 March 2009 Prague, Czech Republic

Outline

- SuperBelle Project
- Computing Requirements of SuperBelle
- Value Weighted Output
- Commercial Cloud Computing – EC2
- Belle MC production
- Implementation of Belle MC on Amazon EC2
- Benchmarks + Costs of Belle MC on EC2

Problems of Standard Model

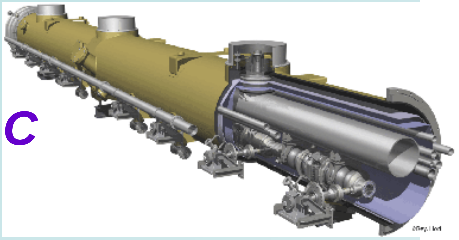
- Quadratic divergence of Higgs mass correction.
  - Indicates new physics in TeV scale.
    - SUSY? Little Higgs? Extra Diemensions
- Pattern of CKM matrix is not explained.
  - Underlining mechanism by new physics implied.
  - Why 3 generations?
- Cannot explain the matter excess of universe.
  - CKM not enough. Other source of CPV implied.
- Does not have dark matter candidates

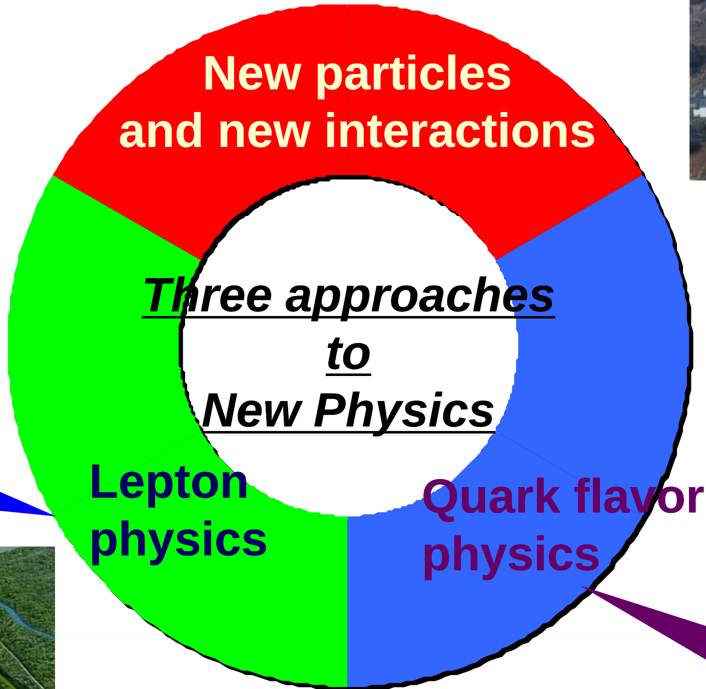
  WMAP : ~1/4 of universe is dark matter
  - Implies new particles (e.g. SUSY has). Extra Dimensions
- Neutrino Masses are too small for Higgs mechanism
  - New Physics at very high Energy

High Energy Physics in the Next Decade

Energy frontier experiments
LHC, ILC, …

Higgs, SUSY, Dark matter,
New understanding of space-time…



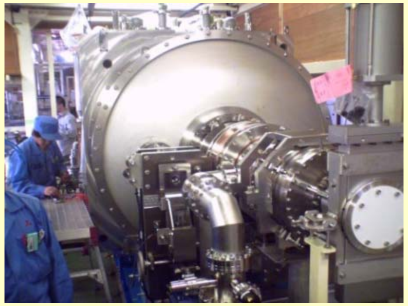


ν exp., μ LFV, τ LFV, g_μ-2, 0νββ …

B Factories, LHCb, K exp., nEDM *etc.*

Neutrino mixing/masses,
Lepton number non-conservation…

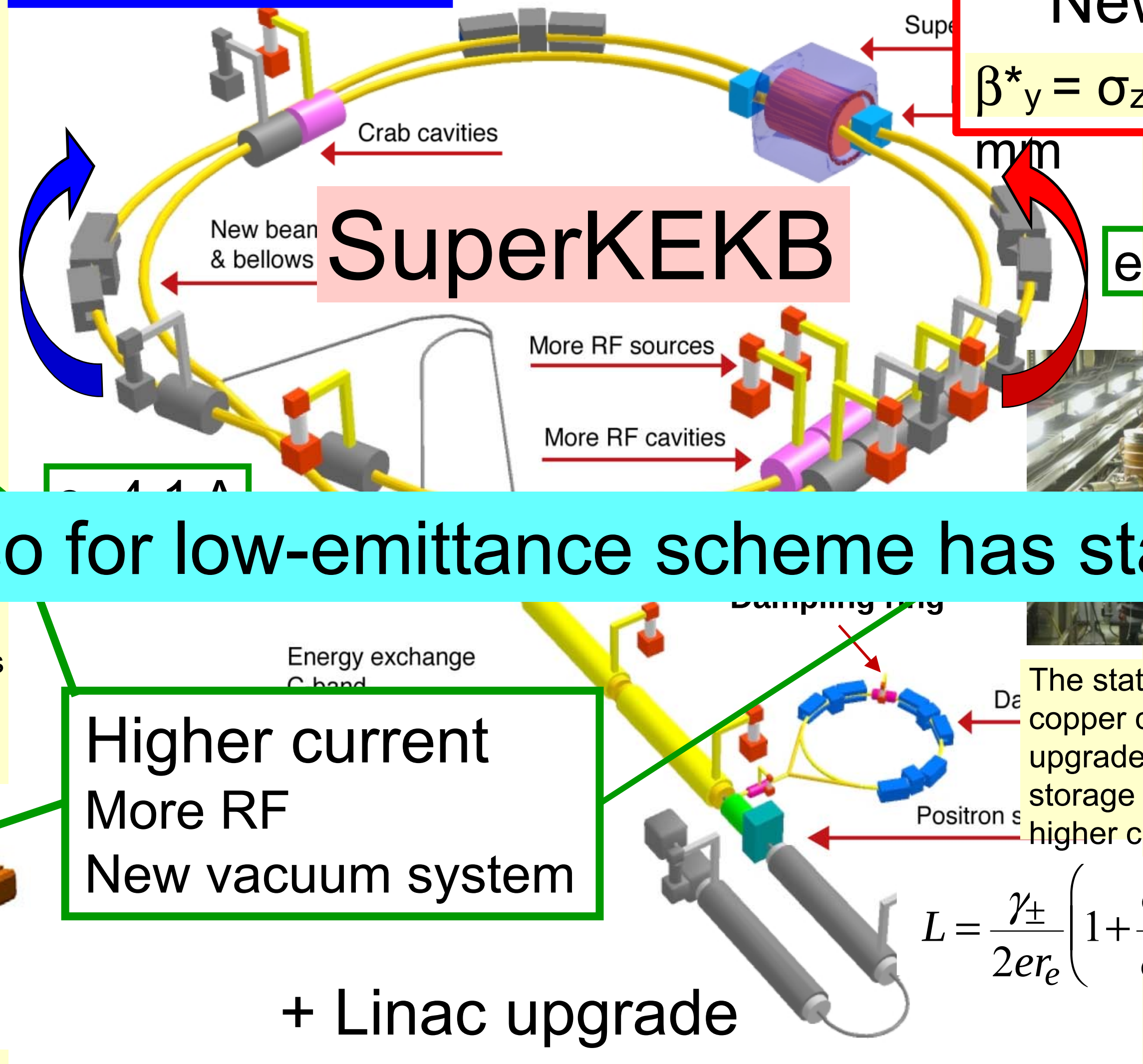
CP asymmetry, Baryogenesis,
Left-right symmetry, New sources of flavor mixing…



Crab cavities installed and undergoing testing in beam

The superconducting cavities will be upgraded to absorb more higher-order mode power up to 50 kW.

Crab crossing



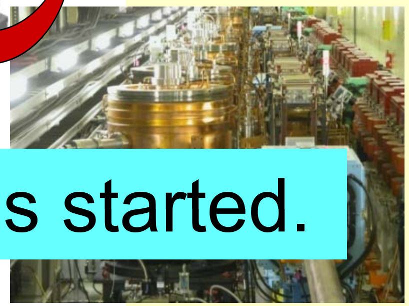
SuperKEKB

New IR

$\beta^*_y = \sigma_z = 3$ mm

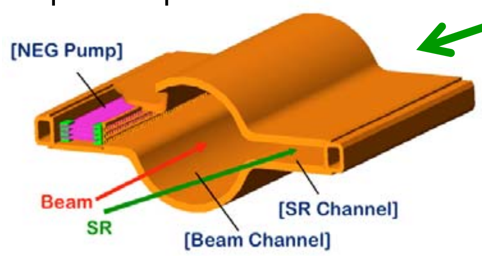
e+ 9.4 A

Studies also for low-emittance scheme has started.



The state-of-art ARES copper cavities will be upgraded with higher energy storage ratio to support higher current.

Higher current
More RF
New vacuum system



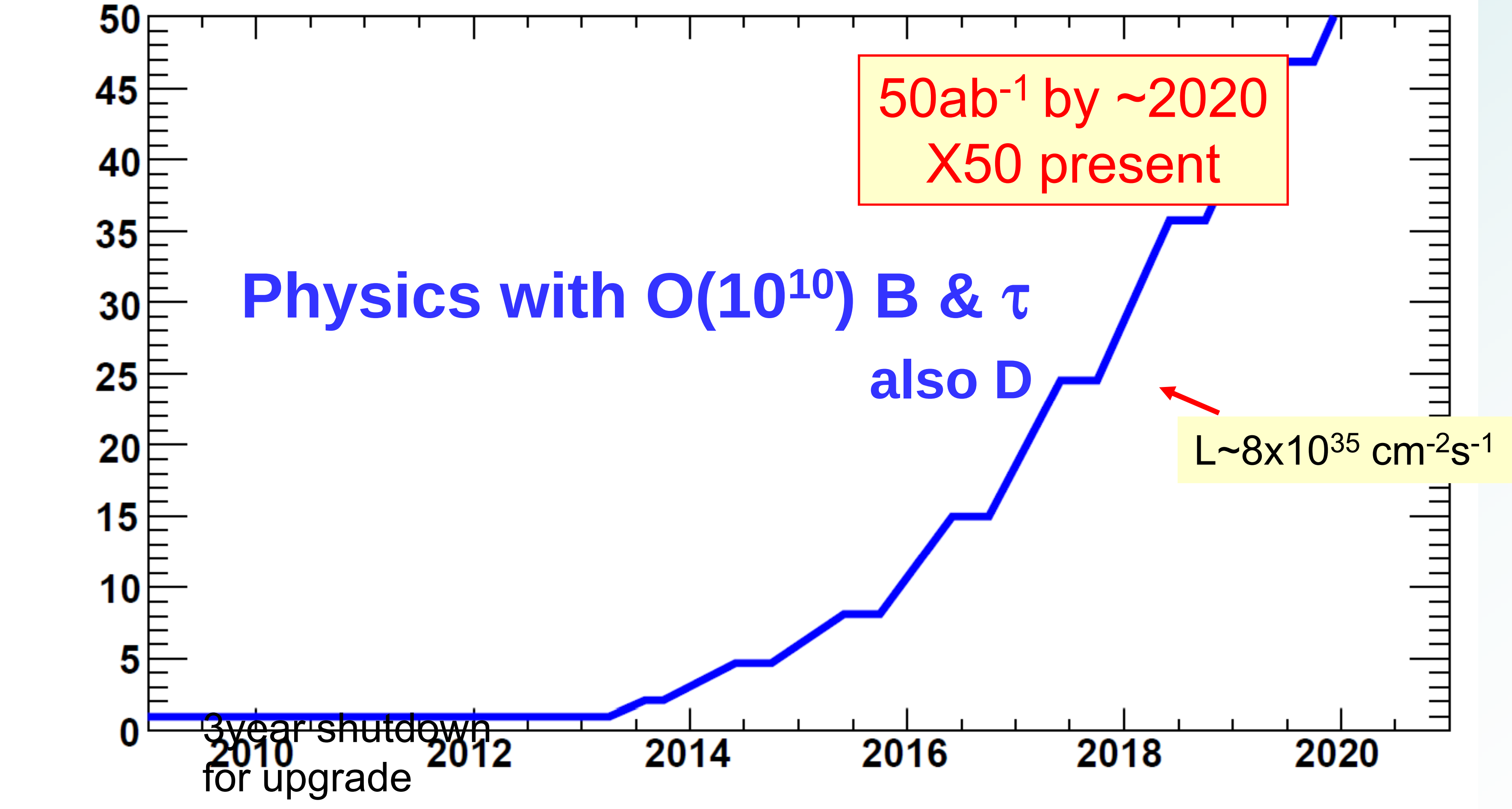
The beam pipes and all vacuum components will be replaced with higher-current design.

+ Linac upgrade

$$L = \frac{\gamma_\pm}{2er_e}\left(1+\frac{\sigma^*_y}{\sigma^*_x}\right)\frac{I_\pm \xi_{\pm y}}{\beta^*_y}\left(\frac{R_L}{R_y}\right)$$

will reach 8 × 10^{35} cm^{-2}s^{-1}

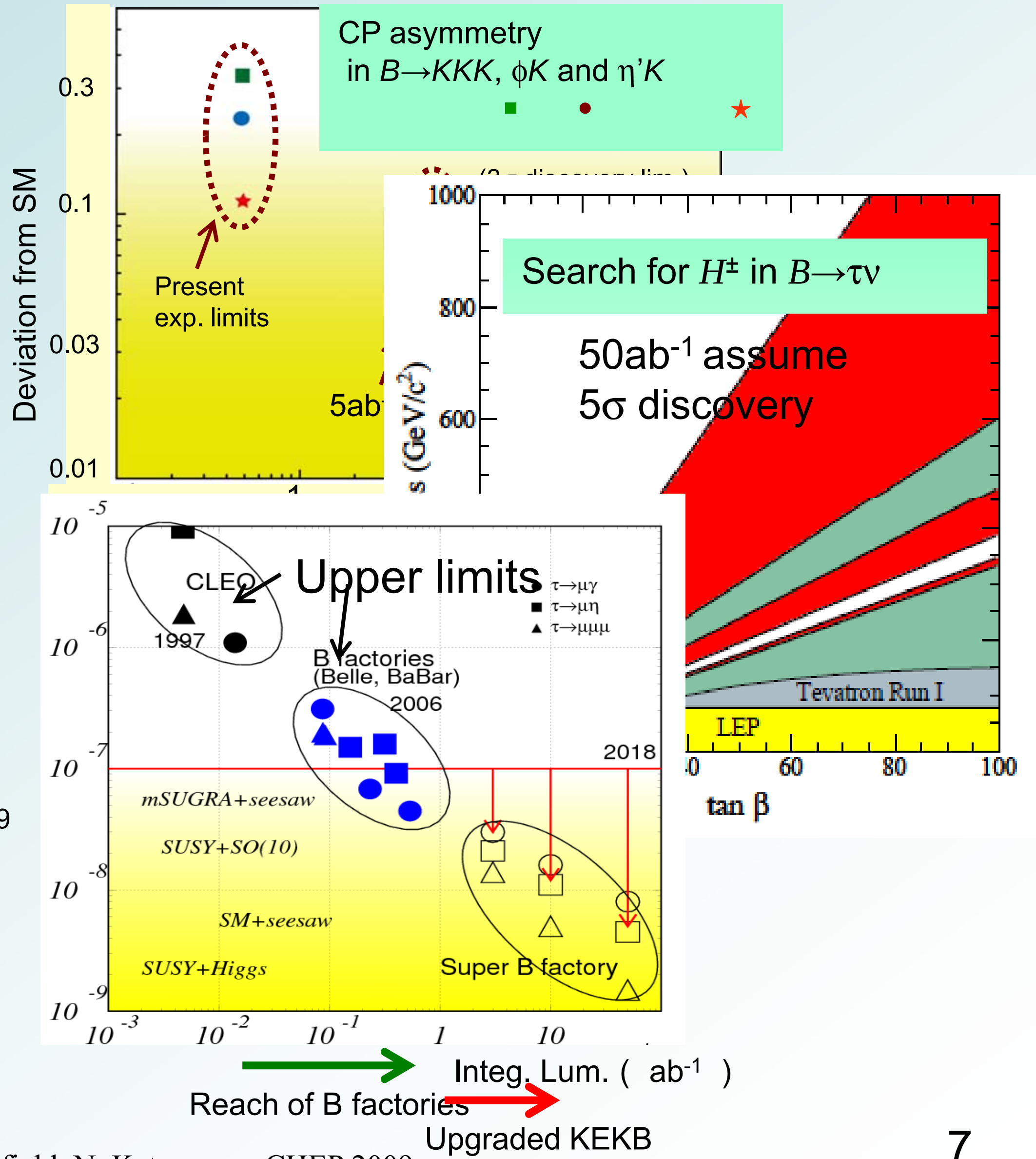
Expected Luminosity



Physics Reach at Super-KEKB/Belle

	Belle'06 (~0.5ab^{-1})	5ab^{-1}	50ab^{-1}
$\Delta S(\phi K^0)$	0.22	0.073	0.029
$\Delta S(\eta' K^0)$	0.11	0.038	0.020
$\Delta S(K_S K_S K_S)$	0.33	0.105	0.037
$\Delta S(K_S \pi^0 \gamma)$	0.32	0.10	0.03
$Br(X_s \gamma)$	13%		
$A_{CP}(X_s \gamma)$	0.058	0.01	0.005
C_9 [$A_{FB}(K^*ll)$]	---	11%	4%
C_{10} [$A_{FB}(K^*ll)$]	---	13%	4%
$Br(B^+ \to K^+ \nu\nu)$	<9Br(SM)	33ab^{-1} for 5σ discovery	
$Br(B^+ \to \tau\nu)$	3.5σ	10%	3%
$Br(B^+ \to \mu\nu)$	<2.4Br(SM)	4.3ab^{-1} for 5σ discovery	
$Br(B^+ \to D\tau\nu)$	---	7.9%	2.5%
$Br(\tau \to \mu\gamma)$	<45	<30	<8
$Br(\tau \to \mu\eta)$	<65	<20	<4
$Br(\tau \to 3\mu)$	<209	<10	<1
$\Delta \sin 2\phi_1$	0.026	0.016	0.012
$\Delta\Phi_2$ ($\rho\pi$)	68°—95°	3°	1°
$\Delta\Phi_3$(Dalitz)	20°	7°	2.5°
ΔV_{ub} (incl.)	7.3%	6.6%	6.1%

X10^{-9}



Funding Status of SuperBelle

- Next Large Project for KEK
- Japanese Government awarded $5 Million for new equipment for SuperBelle for 2009

Current KEKB Computer System

Data size ~ 1 ab^{-1}

New KEK Computer System has 4000 CPU cores

Storage ~ 2 PetaBytes

SuperBelle Requirements

Initial rate of 2×10^{35} $cm^{2} sec^{-1}$ => 4 ab^{-1} /year

Design rate of 8×10^{35} $cm^{2} sec^{-1}$ => 16 ab^{-1} /year

CPU Estimate 10 – 80 times current depending on reprocessing rate

So 4×10^{4} – 3.4×10^{5} CPU cores

Storage 10 PB in 2013, rising to 40 PB/year after 2016

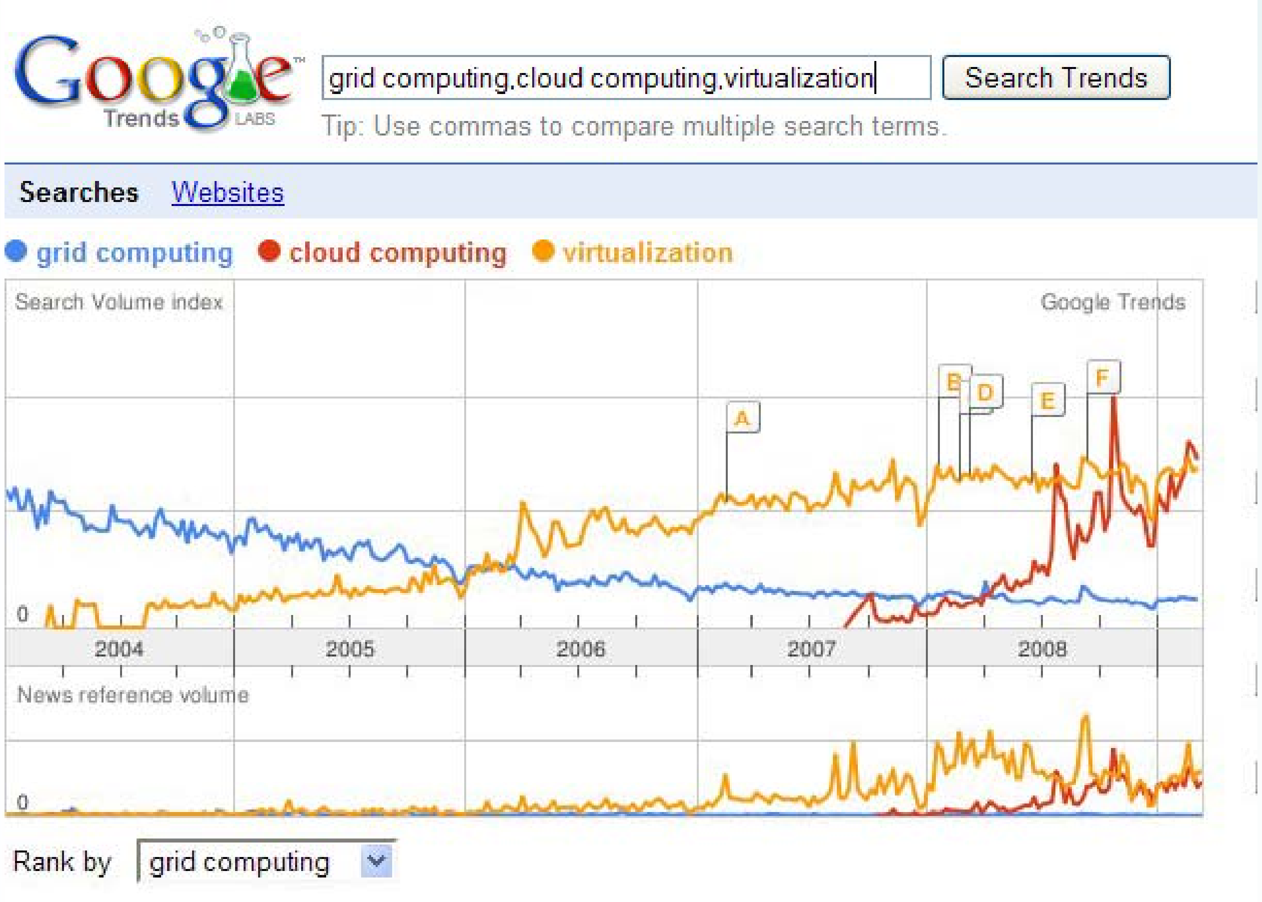
“Cloud Computing”

Cloud Computing has captured market interest

Cloud computing makes large scale computing resources available on a commercial basis

A simple SOAP request creates a “virtual computer” instance with which one can compute as they wish

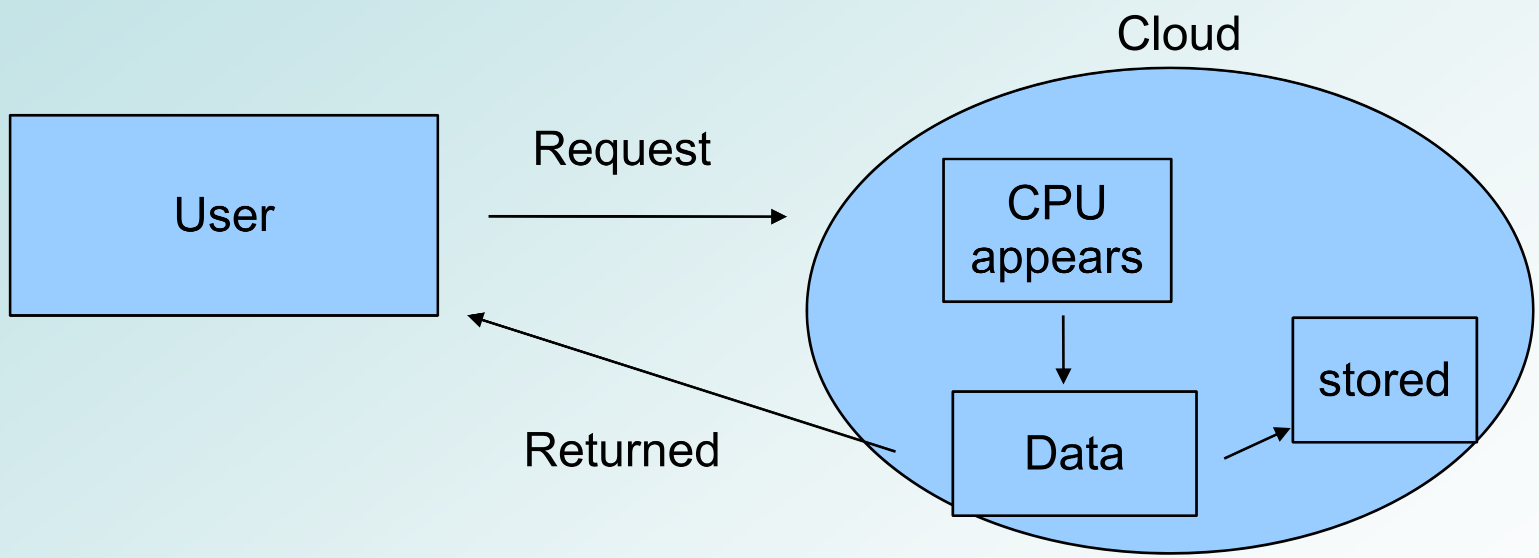
Internet Companies have massive facilities and scale, order of magnitude larger than HEP



Can we use Cloud Computing to reduce the TCO of the SuperBelle Computing?

Cloud Computing

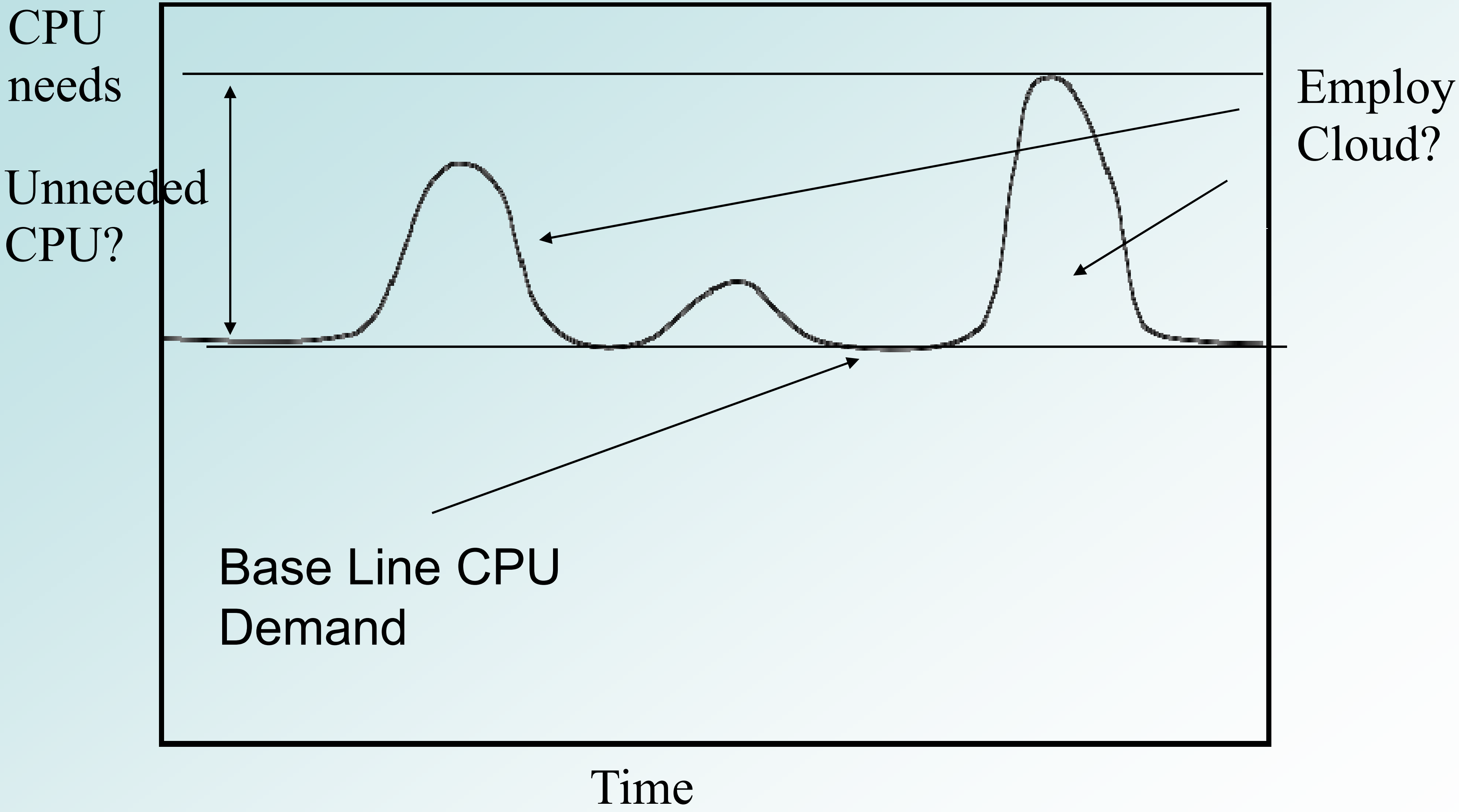
Internet companies have established a Business based on CPU power on demand, one could imagine that they could provide the compute and storage we need at a lower cost than dedicated facilities.



Resources are deployed as needed. Pay as you go.

MC Production is a large fraction of HEP CPU - seems suited to Cloud

Particularly useful for Peak Demand



Value Weighted Output

- Question: Does the value of a cluster decrease with time?
- Yes! We've all seen sad old clusters nobody wants to use.
- How do we quantify how the value of a CPU decreases?
- Moores'Law! "Computing Power Doubles in 1.5 years"

Moores Law: $P = 2^{t/1.5}$ P = CPU Power, t time in years

$\Rightarrow P = e^{\lambda t}$ $\lambda = 0.462$ years^{-1}

Suppose a CPU can produce X events per year at purchase:

Conjecture: The Value of that output drops in proportion to Moores' Law

Define a concept: Value Weighted Output (VWO)

Value Weighted Output, VWO

So for a CPU with an output of X events per year:

$$VWO = Xe^{-\lambda.0} + Xe^{-\lambda.1} + Xe^{-\lambda 2} + Xe^{-\lambda 3} + \ldots$$

Truncating after 5 years (typical lifespan of a cluster), gives

$$VWO = \sum_{t=0}^{5} Xe^{-\lambda.t} \cong \int_{0}^{5} Xe^{-\lambda t} = 1.9X$$

(Taking t to infinity gives 2.2 X)

On the other hand the support costs are constant or increase with time

So another attraction of Cloud Computing is that the purchase of CPU power can
Be made on a yearly basis.

The legacy kit of earlier purchases need not be maintained

Downsides are well known here. Not least of which is Vendor lock in.

Amazon Elastic Computing Cloud (EC2)

- Acronyms For EC2
- Amazon Machine Image (AMI)
  - Virtual Machine employed for Computing
  - ($0.1 - $0.8 per hour of use)
- Simple Storage Solution (S3)
  - ($0.15 per Gb per Month)
- Simple Queuing Service (SQS)
  - Used control Monitor jobs on AMI's via polling (pay per poll)
  - Really cheap!

•Chose to investigate EC2 in detail because it appeared the most mature
•Complete access to AMI as root via ssh.
•Large user community
•Lots of Documentation and online Howto's
•Many additional OpenSource tools

Interfacing with EC2

- Use SOAP requests
- Elastic Fox plugin for Firefox
  - Provided by Amazon
  - Register, locate, view, start and stop AMIs from Firefox
- S3Fox
  - 3rd party firefox plugin
  - Move data to and from S3 with Firefox
- Browser control very helpful getting started with EC2

Download Elastic Fox from:
http://developer.amazonwebservices.com/connect/entry.jspa?externalID=609
Getting Started Guide:
http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1797&categoryID=174
S3Fox
http://www.rjonna.com/ext/s3fox.php

Building the AMI's

- AMI's can be anything you want.
- Many prebuilt AMI's available but no Scientific Linux
- Create Virtual Machines and Store them on S3
- Built 4 AMI's
  - An Scientific Linux (SL) 4.6 instance (Public)
  - SL4.6 with Belle Library (Used in initial Tests) (Private)
  - SL5.2 (Public)
  - SL5.2 with Belle Library (Production, Private)

•We used a loopback block device to create our virtual image.
•Standard yum install of SL but with a special version of tar
•Belle Libraries added to the base AMI's via rpm and yum
•Uploaded to S3 and registered

•(Found a bug in ElasticFox during the SL5.2 registration!)

Playing with AMI's

- Important part is to load a script that pulls your ssh key used to register the AMI from Amazon
- With this you can ssh dirctly into your AMI as root! (no root password)
- Start the AMI with ElasticFox => gets a IP
- ssh as root into the IP => full control!
- scp data to and from the AMI
- Create as many AMIs as you wish to pay for
- ($0.1 - $0.8 per hour)

Initial Tests

- Quick tests to check things and first guess at costs

Instance Type	EC2CU	RAM	ARCH	$/Hour
m1.small	1	1.7	32 bit	0.10
m1.large	4	7.5	64 bit	0.40
m1.xlarge	8	15	64 bit	0.80
c1.medium	5	1.7	32 bit	0.20
c1.xlarge	20	17	64 bit	0.80

Initial Test results

Machine	cost/10^4 events	cost/10^9 events
Small EC2 Instance	\$2.065	\$206,541.575
Large EC2 Instance	\$1.175	\$117,504.489
Extra Large EC2 Instance	\$1.176	\$117,637.111
HighCPU Med EC2 Instance	\$1.029	\$102,913.583
HighCPU XL EC2 Instance	\$0.475	\$47,548.933

10^9 events is approximately the MC requirement of a Belle 3-month run

PowerEdge 1950 8-core box (used in Melbourne Tier 2) Cost ~ \$4000
10^4 events in 32 minutes

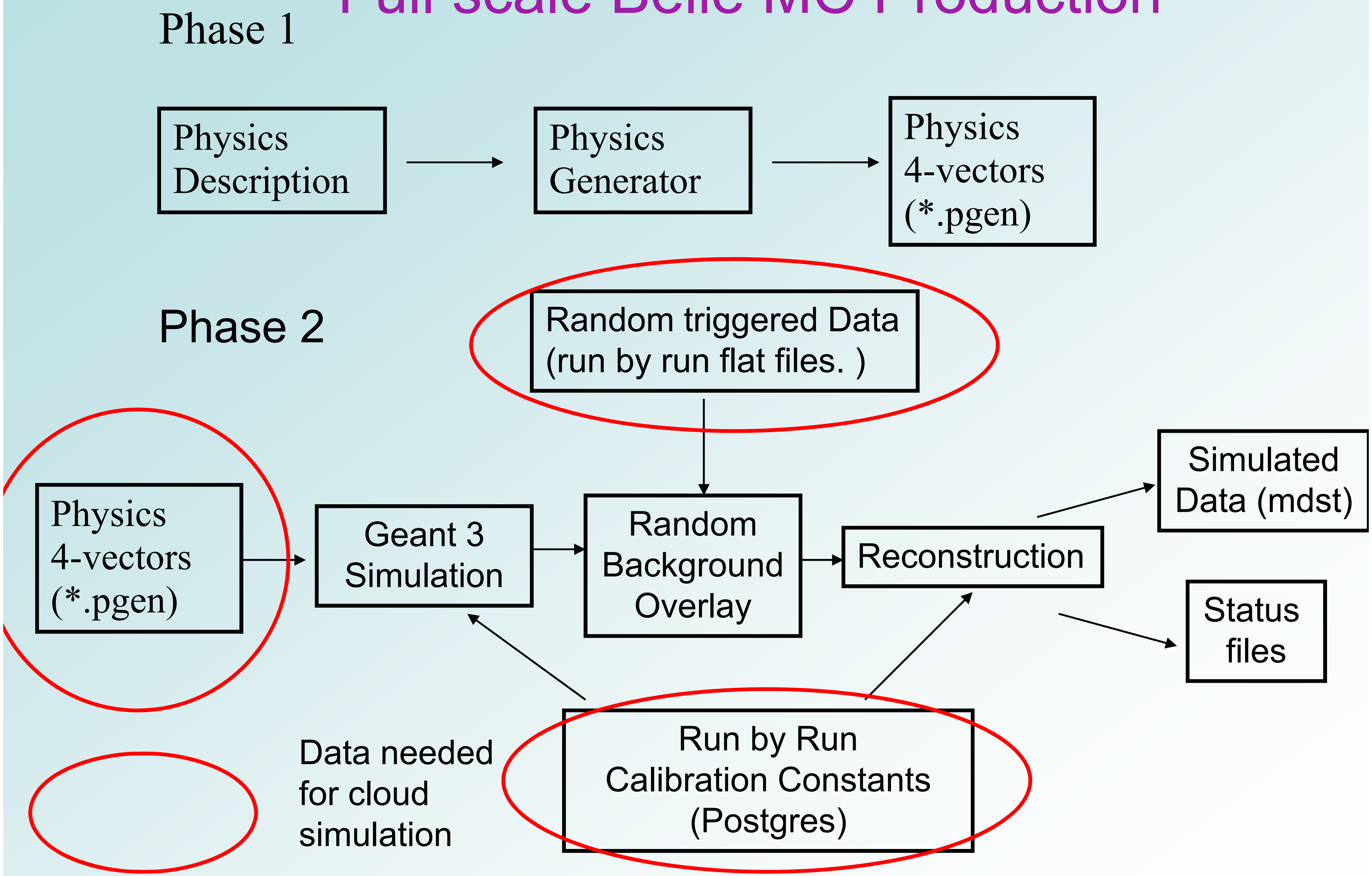
Amortization Period	Events Generated	Cost/10^4 events	VWO Cost/10^4 events
8000 hours - 1 Year	42×10^6 events	\$0.95	\$0.50
16000 hours - 2 Year	84×10^6 events	\$0.47	
24000 hours - 3 Years	126×10^6 events	\$0.32	

Electricity consumption: 400 W => 3500 KWhr/Yr ~\$700/year in Japan
Over 5 years, VWO cost is \$0.94 per 10^4 events

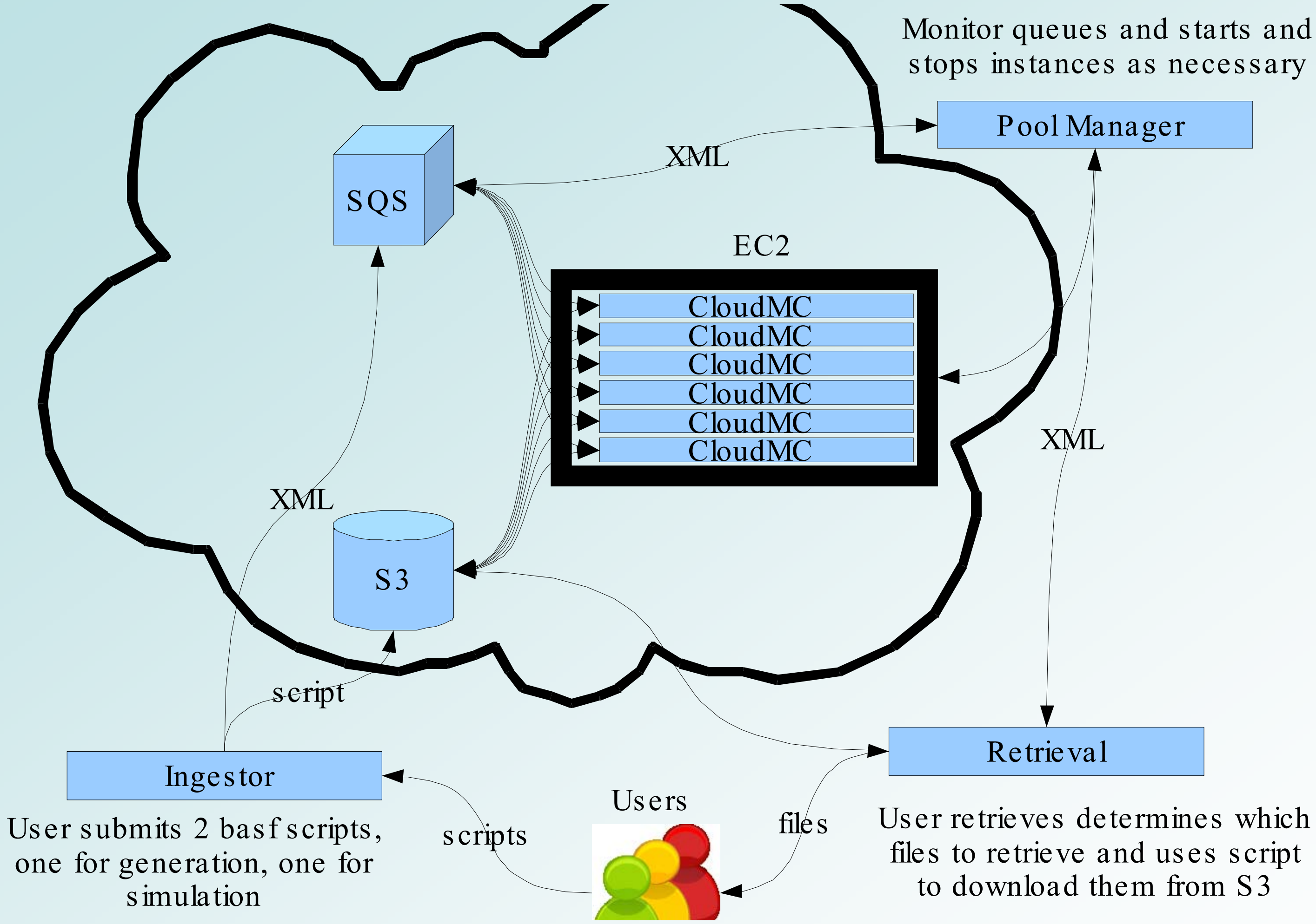
Full scale tests

- Initial test was for a series of runs on a single CPU
- Neglected important additional steps as well as startup/shutdown
- Next step was a full scale Belle MC production test.
- Million event Generation to be used for Belle Analysis

Full scale Belle MC Production



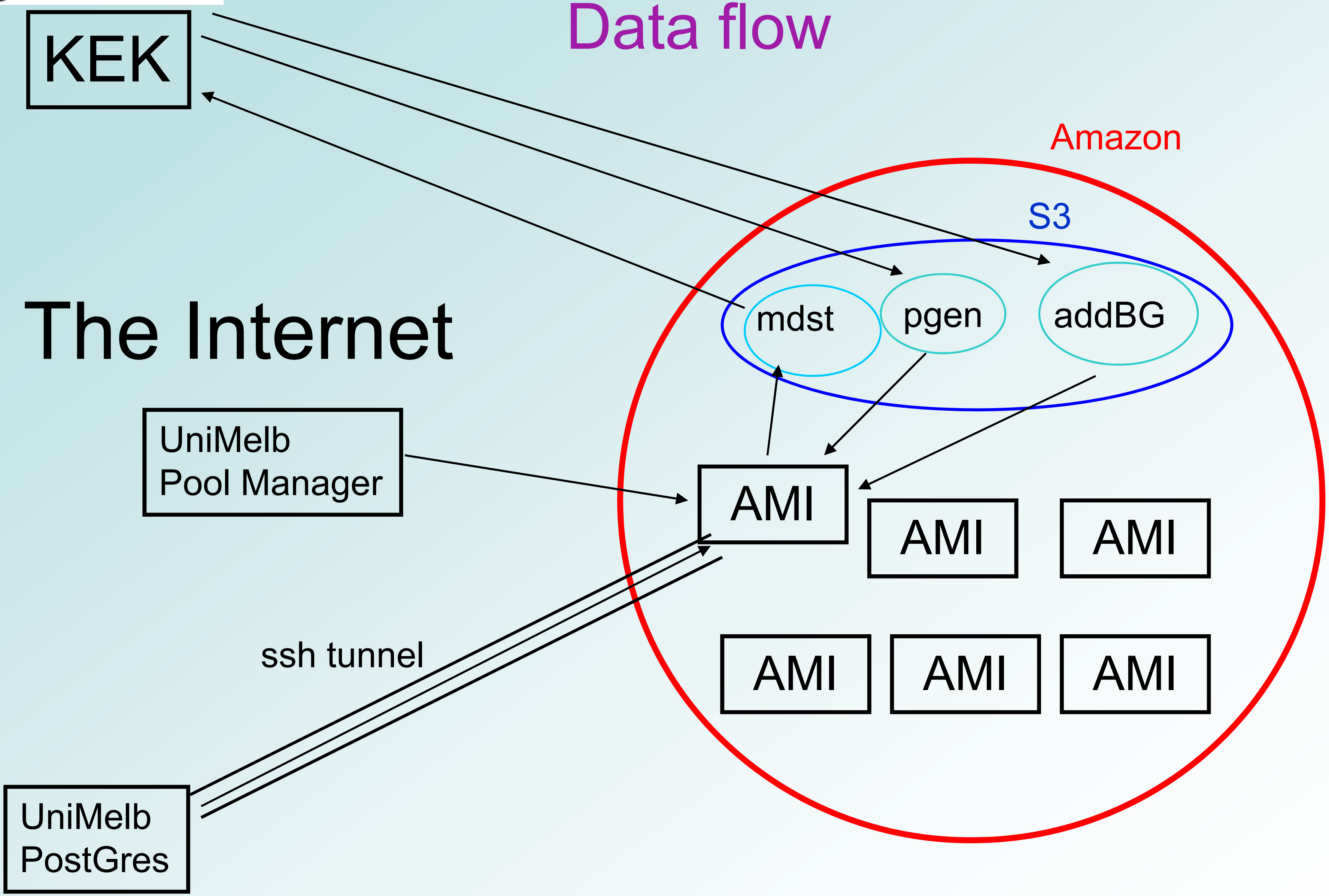
Automating Cloud Production



Accessing Data on the Cloud

- Full scale Belle MC production requires 3 types of data
- *.pgen files which contain the 4-vectors of the Physics processes
- Random triggered background Data, (“addBG”) to be overlayed on the Physics
- Calibration constants for alignment and run conditions
- *.pgen and addBG data were loaded onto S3
- Accessed via a FUSE module and loaded into each AMI instance
- Calibration data was accessed via an ssh tunnel to a postgres server at Melbourne

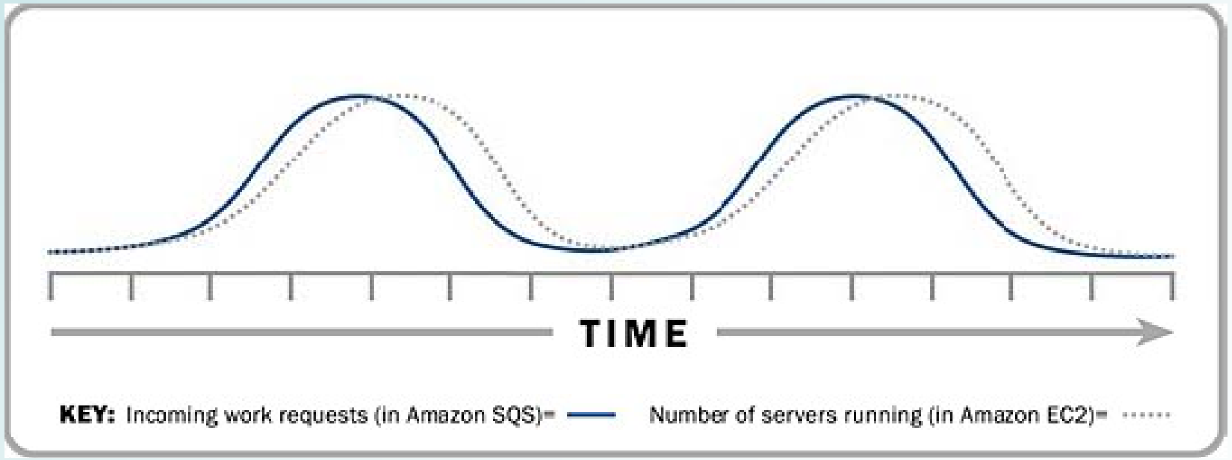
Data flow



KEK
Amazon
S3
mdst
pgen
addBG
The Internet
UniMelb
Pool Manager
AMI
AMI
AMI
AMI
AMI
AMI
ssh tunnel
UniMelb
PostGres

Lifeguard

- Employ an Open Source tool called Lifeguard to manage the pool of AMIs.
- Manages the MC production as a Queuing Service
- Constantly monitors the queue
- Starts and stops AMIs as necessary
- Deals with non-responsive AMIs
- Tracks job status



Shutsdown idle AMI's at the end

Detailed Flow and Timings

- 1. Copy addBG Data to S3
  2. Copy pgen files to S3
- 1. Start AMI Instance [3-10mins]
  2. Download and setup SSH key from special amazon address [1secs]
  3. Download cloud_mc_production package (Melbourne) and untar [2mins]
  4. Download lifeguard package (Melbourne) and untar [3mins]
  5. Execute lifeguard service [0secs]
- On the first job that runs:
  1. Setup an SSH tunnel to the postgres server [5secs]
  2. load s3fs fuse module [1secs]
  3. make mountpoint and mount pgen [2secs]
  4. make mountpoint and mount addbg [2secs]
  5. untar pgen and copy from S3 to local AMI disk [10mins]
  6. copy addbg files from S3 to local AMI disk [10mins]

  (This step implies 5 MB/Sec transfer from S3 to local instance)

Then as below:

- Otherwise:
  1. start basf [5secs]
  2. (basf accesses melbourne postgres server) [1mins]
  3. Start MC generation
  4. Complete MC generation (BASF stops)
  5. Transfer output files (mdst, status only) to S3 [<5mins]

At Completion of the MC block, pull the data from Cloud into KEK

Screenshots

- Ingesting

```
xenon lifeguard #
xenon lifeguard #  ./cloud_sub.sh generation.scr simulation.scr TestCloudSub2 001
Feb 24, 2009 4:21:04 PM com.directthought.lifeguard.ingestor.FileIngestor main
FINE: using bucket :autocloudmc
Feb 24, 2009 4:21:04 PM com.directthought.lifeguard.util.QueueUtil getQueueOrElse
FINE: trying...
Feb 24, 2009 4:21:07 PM com.directthought.lifeguard.util.QueueUtil getQueueOrElse
FINE: trying...
Feb 24, 2009 4:21:11 PM com.directthought.lifeguard.IngestorBase ingest
FINE: ingested:input.sh [E9275C64EAAE8996E1B0600F1BF18550]
```

- Pool Manager starts a new instance

```
INFO: queue:0 servers:0 load:0 ii:50 bi:0
Feb 24, 2009 4:21:08 PM com.directthought.lifeguard.PoolManager run
INFO: queue:0 servers:0 load:0 ii:54 bi:0
Feb 24, 2009 4:21:12 PM com.directthought.lifeguard.PoolManager run
INFO: queue:1 servers:0 load:0 ii:59 bi:0
>>>>>>>>>>> instance started : i-d44ed8bd
Feb 24, 2009 4:21:20 PM com.directthought.lifeguard.PoolManager run
INFO: queue:1 servers:1 load:0 ii:0 bi:0
Feb 24, 2009 4:21:25 PM com.directthought.lifeguard.PoolManager run
INFO: queue:1 servers:1 load:0 ii:4 bi:0
Feb 24, 2009 4:21:29 PM com.directthought.lifeguard.PoolManager run
INFO: queue:1 servers:1 load:0 ii:9 bi:0
```

Screenshots

- 10 jobs in the queue, 9 Instances start

Reservation ID	Instance ID	AMI		State	Pu...	Privat...	Key	...	R...	...	Type	Local Launch...	Availability Z...
r-b069d0d9	i-6823b501	ami-06fc1b6f		running	ec2-1...	domU-12...	fifieldt	default		0	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-6720b60e	ami-06fc1b6f		pending			fifieldt	default		0	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-6620b60f	ami-06fc1b6f		pending			fifieldt	default		1	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7920b610	ami-06fc1b6f		pending			fifieldt	default		2	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7820b611	ami-06fc1b6f		pending			fifieldt	default		3	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7b20b612	ami-06fc1b6f		pending			fifieldt	default		4	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7a20b613	ami-06fc1b6f		pending			fifieldt	default		5	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7d20b614	ami-06fc1b6f		pending			fifieldt	default		6	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7c20b615	ami-06fc1b6f		pending			fifieldt	default		7	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7f20b616	ami-06fc1b6f		pending			fifieldt	default		8	m1.small	2009-02-25 10:4...	us-east-1b

- 10 jobs in the queue, 10 instances run

Reservation ID	Instance D	AMI		State	Pub...	Priva...	Key	...	R...	...	Type	Local Launch. .	Availability Z...
r-b069d0d9	i-6823b501	ami-06fc1b6f		running	ec2-17...	domU-1...	fifield:	default		0	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-6720b60e	ami-06fc1b6f		running	ec2-72...	domU-1...	fifield:	default		0	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-6620b60f	ami-06fc1b6f		running	ec2-17...	domU-1...	fifield:	default		1	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7920b610	ami-06fc1b6f		running	ec2-72...	domU-1...	fifield:	default		2	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7820b611	ami-06fc1b6f		running	ec2-17...	domU-1...	fifield:	default		3	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7b20b612	ami-06fc1b6f		running	ec2-72...	domU-1...	fifield:	default		4	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7a20b613	ami-06fc1b6f		running	ec2-17...	domU-1...	fifield:	default		5	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7d20b614	ami-06fc1b6f		running	ec2-17...	domU-1...	fifield:	default		6	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7c20b615	ami-06fc1b6f		running	ec2-17...	domU-1...	fifield:	default		7	m1.smal	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7f20b616	ami-06fc1b6f		running	ec2-75...	domU-1...	fifield:	default		8	m1.smal	2009-02-25 10:4...	us-east-1b

- Jobs finish, idle timeout expires, shut down

Reservation ID	Instance ID	AMI		State	Pub...	Priva...	Key	...	Re...		Type	Local Launch...	Availability Z...
r-b069d0d9	i-6823b501	ami-06fc1b6f		shuttin...	ec2-17...	domU-1...	fifieldt	default	User in...	0	m1.small	2009-02-25 10:4...	us-east-1b
r-9468d1fd	i-9c20b6f5	ami-06fc1b6f		shuttin...	ec2-17...	domU-1...	fifieldt	default	User in...	0	m1.small	2009-02-25 10:5...	us-east-1b
r-3e68d157	i-6620b60f	ami-06fc1b6f		shuttin...	ec2-17...	domU-1...	fifieldt	default	User in...	1	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7920b610	ami-06fc1b6f		shuttin...	ec2-72...	domU-1...	fifieldt	default	User in...	2	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7820b611	ami-06fc1b6f		shuttin...	ec2-17...	domU-1...	fifieldt	default	User in...	3	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7b20b612	ami-06fc1b6f		shuttin...	ec2-72...	domU-1...	fifieldt	default	User in...	4	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7a20b613	ami-06fc1b6f		shuttin...	ec2-17...	domU-1...	fifieldt	default	User in...	5	m1.small	2009-02-25 10:4...	us-east-1b
r-3e68d157	i-7d20b614	ami-06fc1b6f		shuttin...	ec2-17...	domU-1...	fifieldt	default	User in...	6	m1.small	2009-02-25 10:4...	us-east-1b

Retrieval

- Using the ID from ingest, list & retrieve files produced

```
xenon lifeguard # ./retrieve.pl -i 5A1A2F43BE25F0B34BC22EBB262804D8 -l
CloudMC-001-TestCloudSub2 input: 5A1A2F43BE25F0B34BC22EBB262804D8
2DFC915CFA8BF9868279F2146C9B558F TestCloudSub2-001-service.log
C6B07F632F24BE1E4E692062BE67BA42 TestCloudSub2-001-generation.log
A42BACD8DC3B348E18D01906198ACC91 TestCloudSub2-001-simulation.log
B8294F73D44054F1B8091D10A151489A TestCloudSub2-001-template.mdst
xenon lifeguard # ./retrieve.pl -i 5A1A2F43BE25F0B34BC22EBB262804D8 C6B07F632F24BE1E4E692062BE67BA42 A42BACD8DC3B348E18D01906198ACC91 B
3D44054F1B8091D10A151489A
```

Results

- Exp 61 Charged RunRange 3 Stream 1 ~ 850 K events
- Run 20 HighCPU-XL instances (8 cores, 17Gb RAM)
- Retrieve addbg data from S3
- Store results in S3 before transfer to KEK
- A way to look at real cost of cloud

Running

Reservation ID	Instance ID	AMI	State	Public DNS	Private DNS	Key	Type	Local Launch Time	Tag
-fd46c194	i-ae71efc7	ami-7735d...	terminated			fifieldt	c1.xlarge	2009-03-15 14:22:32	
-fd46c194	i-a171efc8	ami-7735d...	terminated			fifieldt	c1.xlarge	2009-03-15 14:22:32	
-e940c780	i-1169f778	ami-7735d...	terminated			fifieldt	c1.xlarge	2009-03-15 16:16:03	
-aa40c7c3	i-d669f7bf	ami-7735d...	running	ec2-174-129-116-137.compu...	ip-10-250-163-223.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-a969f7c0	ami-7735d...	running	ec2-75-101-243-21.compute-...	ip-10-250-43-159.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-a869f7c1	ami-7735d...	running	ec2-75-101-226-223.comput...	ip-10-250-94-207.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-ab69f7c2	ami-7735d...	running	ec2-67-202-38-36.compute-1...	ip-10-250-222-239.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-aa69f7c3	ami-7735d...	running	ec2-174-129-114-89.comput...	ip-10-251-63-239.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-ac69f7c5	ami-7735d...	running	ec2-72-44-33-190.compute-1...	ip-10-250-230-255.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-af69f7c6	ami-7735d...	running	ec2-174-129-153-242.compu...	ip-10-250-71-63.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-ae69f7c7	ami-7735d...	running	ec2-75-101-205-44.compute-...	ip-10-250-198-239.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-a169f7c8	ami-7735d...	running	ec2-67-202-1-207.compute-1...	ip-10-250-146-223.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:19:23	
-aa40c7c3	i-ad69f7c4	ami-7735d...	terminated			fifieldt	c1.xlarge	2009-03-15 16:19:23	
-574fc83e	i-2766f84e	ami-7735d...	running	ec2-174-129-156-115.compu...	ip-10-250-211-79.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:29:30	
-164fc87f	i-dc66f8b5	ami-7735d...	pending			fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-c566f8ac	ami-7735d...	running	ec2-75-101-216-251.comput...	ip-10-250-210-175.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-c466f8ad	ami-7735d...	running	ec2-75-101-217-173.comput...	ip-10-250-35-175.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-c766f8ae	ami-7735d...	running	ec2-72-44-36-91.compute-1....	ip-10-250-34-47.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-d966f8b0	ami-7735d...	running	ec2-174-129-182-249.compu...	ip-10-250-187-207.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-d866f8b1	ami-7735d...	running	ec2-75-101-243-99.compute-...	ip-10-250-30-159.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-db66f8b2	ami-7735d...	running	ec2-75-101-181-136.comput...	ip-10-250-127-143.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-da66f8b3	ami-7735d...	running	ec2-174-129-178-75.comput...	ip-10-250-63-239.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-dd66f8b4	ami-7735d...	running	ec2-67-202-16-241.compute-...	ip-10-250-218-95.ec2.internal	fifieldt	c1.xlarge	2009-03-15 16:35:35	
-164fc87f	i-c666f8af	ami-7735d...	terminated			fifieldt	c1.xlarge	2009-03-15 16:35:35	

Costs

- Managed to do 752,233 events in time for this presentation
- CPU cost: $80
  - 20 Instances, 4 hours 57minutes
- Storage cost: $0.20
  - Storage on S3: Addbg 3.1Gb, pgen 0.5Gb, results 37Gb, $6.08/month or $0.20/day
- Transfer cost: $6.65
  - Addbg, pgen in: $0.36, mdst out: $6.29
- **Total Cost: $86.85**

Naïve early estimate without automation and storage overhead ~$40

Need to get equivalent times for GRID production of MC data

Conclusions

- Value Weighted Output – metric to estimate the present time value of CPU
- Can make a few more tweaks to minimize costs
- Charged for the period of time we claim the AMIs
- Keep AMIs active!
- Cloud is promising for MC production
- Can deliver Peak Demand if needed
- Transfer speeds from S3 to AMI likely too slow for large scale data analysis for HEP
- On-demand creation of virtual machines is a flexible way of utilizing Computational Grids

Thank you!