



Dynamic Grid Computing  
With Red Hat Enterprise MRG & Amazon EC2  
Bryan Che  
Product Manager, Red Hat

# Challenges and Requirements for Today's Large-Scale Computing

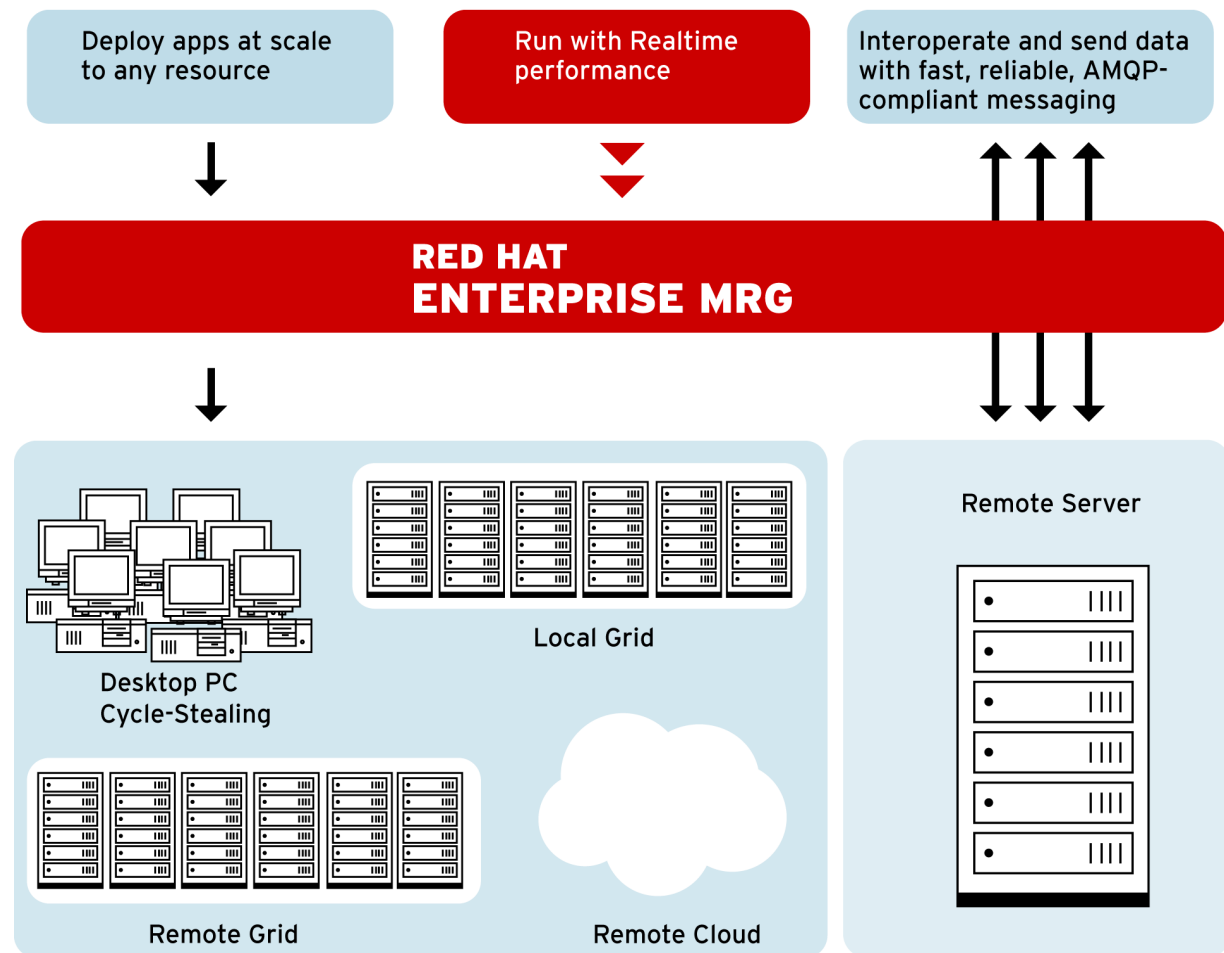
- Most enterprises are running more and more distributed applications and workloads and dealing with an increasing amount of distributed data
- Many enterprises have increasing demands for computing power while simultaneously needing to cut down on physical resource consumption (e.g. power, space)
- Many enterprises have variable needs for their amounts of computing power
  - Virtualization and consolidation can only get you so far if you have significant variability in computing requirements or server loads
  - Some enterprises can never get enough computing capacity

# The Promises and Challenges of Cloud Computing

- Many companies are looking at a variety of cloud-based offerings to address their IT Infrastructure challenges
  - Research: Folding@Home, BOINC, Fedora Nightlife
  - Enterprise: Amazon EC2, Google App Engine
- There are important issues to consider when evaluating cloud infrastructure, including:
  - Do I need to integrate the services with anything local in my datacenter or organization?
  - Do I have any particular requirements about data security?
  - How large are my datasets?
  - Are there any issues with platform lock-in?
  - Do I have any requirements or SLA's for how quickly my results return?
  - Do I want to add capacity to my IT infrastructure or outsource my IT infrastructure?

## About Red Hat Enterprise MRG

- Integrated platform for high performance distributed computing
- High speed, interoperable, open standard **Messaging**
- Deterministic, low-latency **Realtime** kernel
- High performance & throughput computing **Grid** scheduler for distributed workloads,



## About MRG Grid

- Brings advantages of scale-out and flexible deployment to any application
- Delivers better asset utilization, allowing applications to take advantage of all available computing resources
- Dynamically provisions additional peak capacity for “Christmas Rush”-like situations
- Executes across multiple platforms and in virtual machines
- Schedules from sub-second jobs to long-running batch jobs
- Provides seamless and flexible High Throughput Computing (HTC) and High Performance Computing (HPC) across
  - Local grids
  - Remote grids
  - Remote clouds (Amazon EC2)
  - Cycle-stealing from desktop PCs

## MRG Grid is Based on Condor

- MRG Grid is based on the Condor Project created and hosted by the University of Wisconsin, Madison
- Red Hat and the University of Wisconsin have signed a strategic partnership around Condor:
  - University of Wisconsin makes Condor source code available under OSI-approved open source license
  - Red Hat & University of Wisconsin jointly fund and staff Condor development on-campus at the University of Wisconsin
- Red Hat and the University of Wisconsin's partnership will:
  - Add enhanced enterprise features, management, and supportability to Condor and MRG Grid
  - Add High Throughput Computing capabilities to Linux



## Red Hat is Initially Adding to Condor:

- Enterprise Supportability
  - Break out Condor from statically-linked blob to multiple well-maintained and individually patchable rpm's
- Web-Based Management Console
  - Unified management across all of MRG for job, system, and workload management/monitoring
- Virtualization Support via libvirt Integration
  - Support scheduling of virtual machines on Linux using libvirt API's
- AMQP Messaging Integration
  - Enable job submission to Condor via AMQP Messaging clients
  - Enable sub-second, low-latency scheduling for sub-second jobs
- Amazon EC2 Integration
  - Enable automatic EC2 provisioning, job submission, results storage, teardown via Condor scheduler
  - Runs as a job, so it can be a dependency for other jobs or executed based on rules (e.g. add capacity in EC2 if local grid out of capacity)

# Amazon Web Services Are...



Building block services that allow organizations to innovate and save money

- **Infrastructure As a Service**
  - Amazon Simple Storage Service
  - Amazon Elastic Compute Cloud
  - Amazon Simple Queue Service
  - Amazon SimpleDB
- **Payments As a Service**
  - Amazon Flexible Payments Service
  - Amazon Fulfilment Service
- **People As a Service**
  - Amazon Mechanical Turk
- **Alexa Web Services**
  - Alexa Web Information Service
  - Alexa Top Sites
  - Alexa Site Thumbnail
  - Alexa Web Search Platform

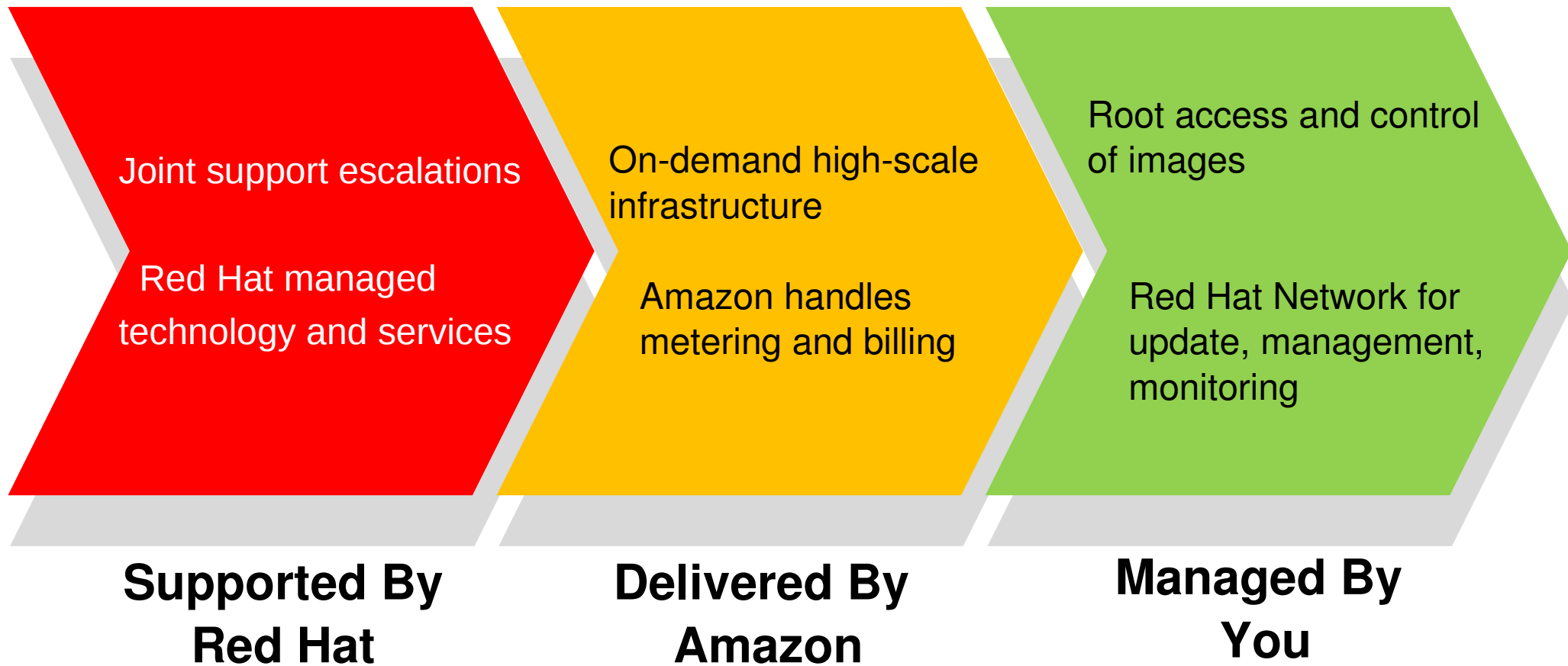
# **Cloud Computing with Red Hat Enterprise Linux - BETA**

## **Red Hat Enterprise Linux...**

- ...Purchase by the hour**
- ...Delivered in the Amazon Web Services cloud**
- ...Supported by Red Hat**

# Red Hat - Amazon Partnership:

Teamed to build complete on-demand IT Infrastructure as a Service



## Fitting the Pieces Together



Amazon Web Services (AWS)

# MRG and Amazon EC2: The Most Flexibility and Best of all Worlds for Grid Deployments

- Transparently move between or blend local grids with cloud-based grids
  - Red Hat Enterprise Linux application certifications hold whether local or in Amazon EC2
  - Red Hat Enterprise MRG provides a unified management interface across local grids, remote grids, and the cloud
- Dynamically add capacity in EC2 to dedicated grids where there is a need to maintain local infrastructure but there is a desire for flexible capacity
  - e.g. Financial Services, Government, Healthcare
- Schedule entirely in EC2 through MRG when you only have occasional needs for grids. Easily add local capacity to your MRG Grid when necessary
  - e.g. Internet, Manufacturing, Research
- No lock-in to local or cloud infrastructure

## MRG and Amazon EC2 Demo Video



## How to Schedule From MRG to Amazon EC2

- MRG Grid will manage the starting, monitoring, and cleaning up of EC2 resources
- Condor jobs on Amazon EC2 are Amazon Machine Images (AMI's)
- Amazon EC2 jobs can run as sub-jobs
  - Automatically add capacity in EC2 when local grid is fully utilized
  - Chain jobs together
- To Schedule Condor jobs on EC2:
  - You:
    - Configure Condor to use EC2
    - Create an AMI, upload it to S3, register it with EC2
    - Create a job description and condor\_submit it
  - Condor will:
    - Start AMI in EC2
    - retrieve user data
    - transfer input
    - run job
    - transfer output
    - shutdown AMI
    - Cleanup EC2 resources
    - Remove job from schedd

## Configure Condor to use EC2

- edit `/etc/condor/condor_config`, which now has settings for Amazon EC2:

```
##  
## EC2: Universe = Grid, Grid_Resource = Amazon  
##  
## The location of the amazon-gahp programs, required  
AMAZON_GAHP = $(SBIN)/amazon-gahp  
AMAZON_GAHP_WORKER = $(SBIN)/amazon-gahp_worker_thread  
## The location of the amazon-gahp EC2 interface script,  
## condor_amazon.pl, required  
AMAZON_EC2_LIB = $(LIBEXEC)  
## Location of log files, useful for debugging, must be in  
## a directory writable by any user, such as /tmp  
AMAZON_GAHP_LOG = /tmp/AmazonGahpLog.$(USERNAME)  
AMAZON_GAHP_WORKER_THREAD_LOG = /tmp/AmazonGahpWorkerLog.$(USERNAME)  
GRIDMANAGER_JOB_PROBE_INTERVAL = 5
```

# Create, Modify, and Register an AMI

- Start an Amazon AMI based on standard Red Hat Enterprise Linux

```
# ec2-run-instances ami-5bae4b32 -k gsg-keypair  
INSTANCE i-10a64379 ami-5bae4b32 pending 0 m1.small 2007-07-11T16:40:44+0000
```

- Customize this AMI with the software you want to run in EC2

```
[log into your new AMI instance]  
[add software to run and add it to /etc/rc.local to launch at AMI startup]  
[add "shutdown -h -P now" to end of job to shutdown AMI]  
...  
# ec2-bundle-vol -d /mnt -k /mnt/pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem -c /mnt/cert-  
HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem -u 495219933132 -r i386  
# ec2-upload-bundle -b <your-s3-bucket> -m /mnt/image.manifest.xml -a <aws-access-key-id> -  
s <aws-secret-access-key>
```

- Register your customized AMI with Amazon EC2

```
# ec2-register <your-s3-bucket>/image.manifest.xml  
IMAGE ami-5bae4b32
```

# Create and Submit Your Condor Job

- Create a job description file for your job (e.g. `my_amazon_ec2_job_file`):

```
# This job targets Amazon EC2
universe = grid
grid_resource = amazon
# A name for the job
executable = my_amazon_ec2_job
transfer_executable = false
# Access keys, needed for authentication
amazon_public_key = AccessKeyID
amazon_secret_key = SecretAccessKey
# The AMI you plan to start as the job
amazon_ami_id = ami-a1b2c3d4
# Some data you pass the AMI
# e.g. the hostname/ip of the central
# manager where you want the instance to
# join as an execute resource
amazon_user_data = demobox.redhat.com
# Queue 10 of these jobs, which means
# 10 instances in EC2
queue 10
```

- Submit your job:

```
# condor_submit my_amazon_ec2_job_file
Submitting job(s)con.
```

- Sample job in AMI's `/etc/rc.local`:

```
# Retrieve "user-data". EC2 stores this information at
# a special URL, which is customized for each running
# instance
USER_DATA=`curl http://169.254.169.254/2007-08-29/user-
data`
# Process USER_DATA, maybe it's format is:
# <command line arguments>;<results file name>
ARGUMENTS="${USER_DATA%;*}"
RESULTS_FILE="${USER_DATA#*;}]"
# Execute the job passing it $ARGUMENTS
/bin/my_program "$ARGUMENTS" > /tmp/output.txt
# Store the results in Amazon S3
/usr/local/s3/s3cmd put /tmp/output.txt
s3://<mybucket>/output.txt
# Shutdown so Condor knows job is completed
shutdown -h -P now
```

## MRG Grid Features

- Management Tools
- Desktop Cycle-Stealing
- Cloud scheduling (Amazon EC2)
- AMQP Messaging Integration – sub-second, messaging API for job submission
- Virtualization – submit a (VM) as a user job; supports migration of the VM
- Policies
- Federated Grids/Clusters
- Multiple Standards-Based APIs
- Workflow Management
- High Availability
- Disk Space Management
- Database Support
- Compute On-Demand
- Dynamic Pool Creation
- Priority Based Scheduling
- Accounting
- Security
- Parallel Universe - extensible framework for running parallel (including MPI) jobs
- Java Universe
- Time Scheduling for Job Execution
- Backfill
- File Staging
- Dedicated and Undedicated Node Management
- Master-Worker (MW) - a single master process can allocate and manage multiple worker processes
- Condor-C – move jobs across queues

# Red Hat Enterprise MRG & EC2 Availability

- MRG v1.0: Available Now; launched at the Red Hat Summit 6/19/2008
  - Realtime and Messaging fully supported
  - MRG Grid is Technology Preview
- MRG v1.1: Available Second Half 2008
  - MRG Grid fully supported
- Cloud Computing with Red Hat Enterprise Linux and Amazon EC2 Beta Service: Available now
  - Provision full Red Hat Enterprise Linux servers on Amazon EC2
  - Pricing starts at \$19/month plus \$0.21 per hour for every deployed server, plus additional bandwidth and storage fees

## Additional Information

- Red Hat Enterprise MRG: <http://redhat.com/mrg>
- Red Hat Enterprise Linux on Amazon EC2 Cloud Service Beta: <http://www.redhat.com/solutions/cloud/>
- Amazon EC2: <http://aws.amazon.com/ec2>



redhat®