Microsoft Research

Enabling Scalable Genomics Research across Desktop and the Cloud

Yogesh Simmhan eScience Group | MSR

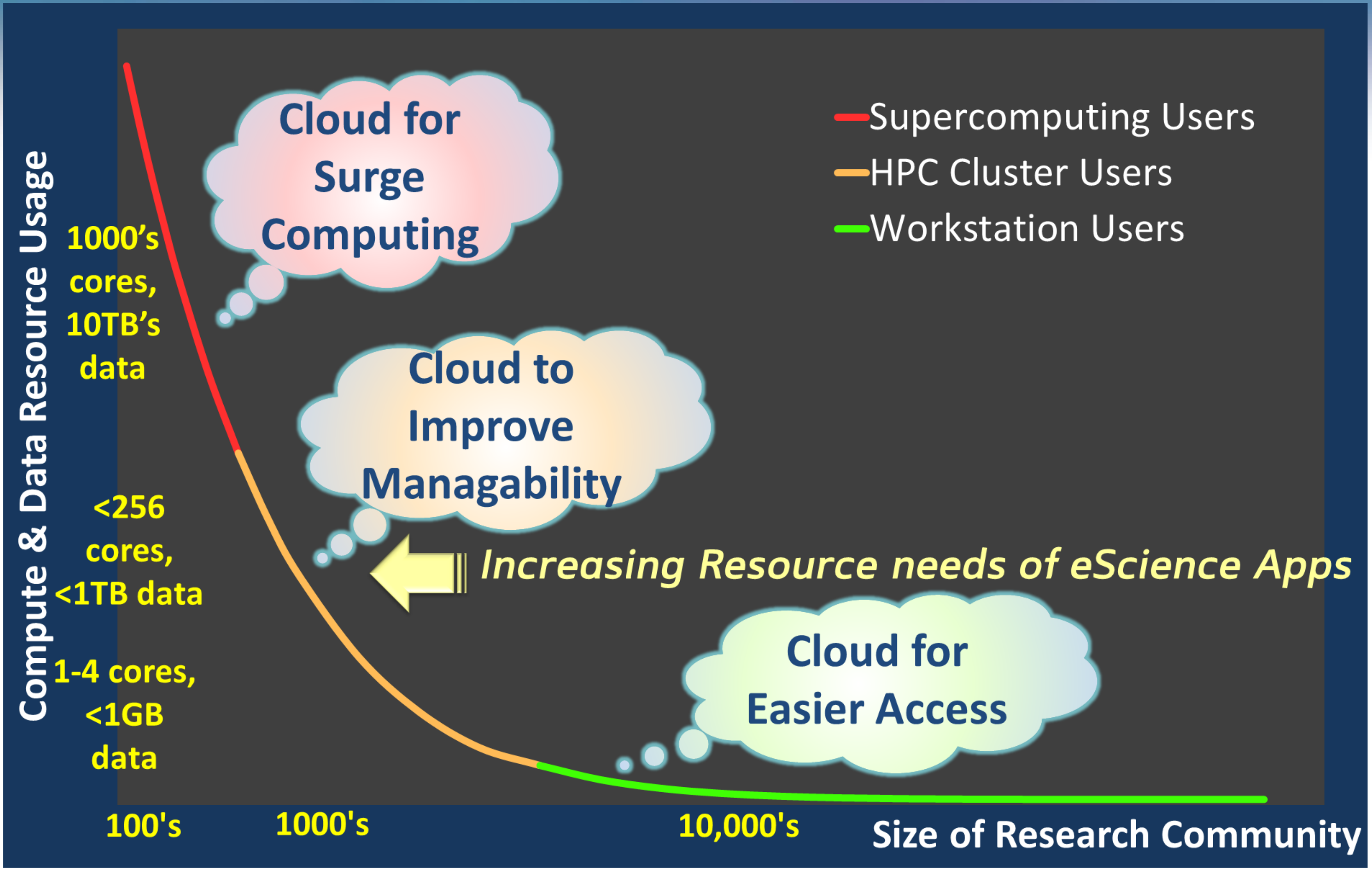
Girish Subramanian DDE Lab | Indiana University
Jennifer Listgarten, David Heckerman, Simon Mercer
& Michael Zyskowski External Research | MSR
Wei Lu Cloud Comp. Futures | MSR

eScience

Increasing eScience Resource Needs

- Researchers confronted with ***surfeit of scientific data***
  - Sensors, shared instruments, simulations
- ***Resource needs*** pushed to the next level
  - Collect, process, analyze, visualize data
- ***Variable workloads*** over time
  - Field campaigns, human subject studies
- Distributed ***Collaborations***
  - Share, track, archive, reuse data

Cloud Opportunities for Science Users

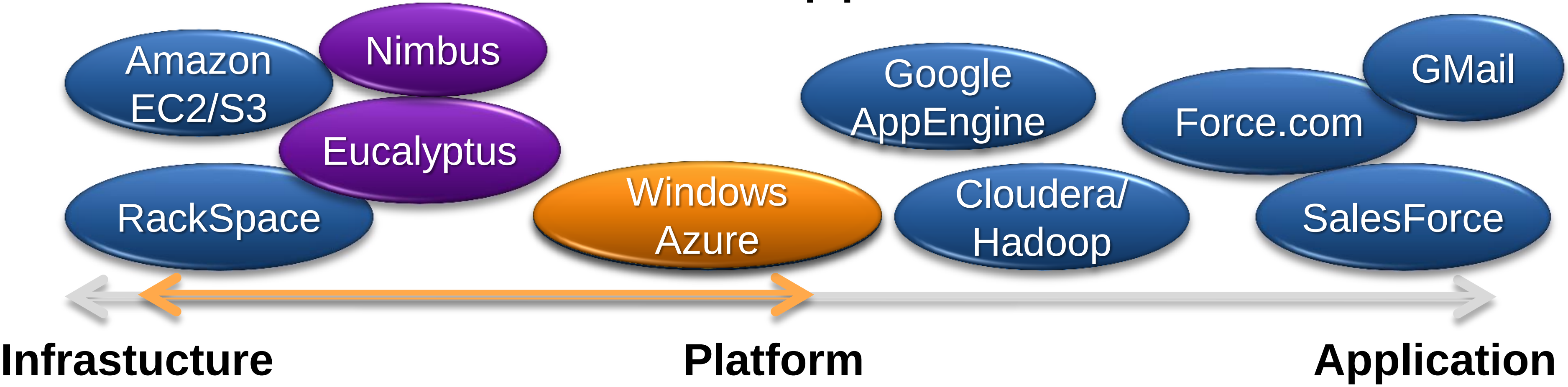


Cloud Computing Advantages

- On-demand ***availability*** of computation & storage
- ***Scalable*** resources from 1 – 100's of nodes
- Ease of resource ***management***
- ***Economical***, Pay as you go, economies of scale, resources upgraded
- ***Simple*** interface using REST Web service

Common Cloud Architectures

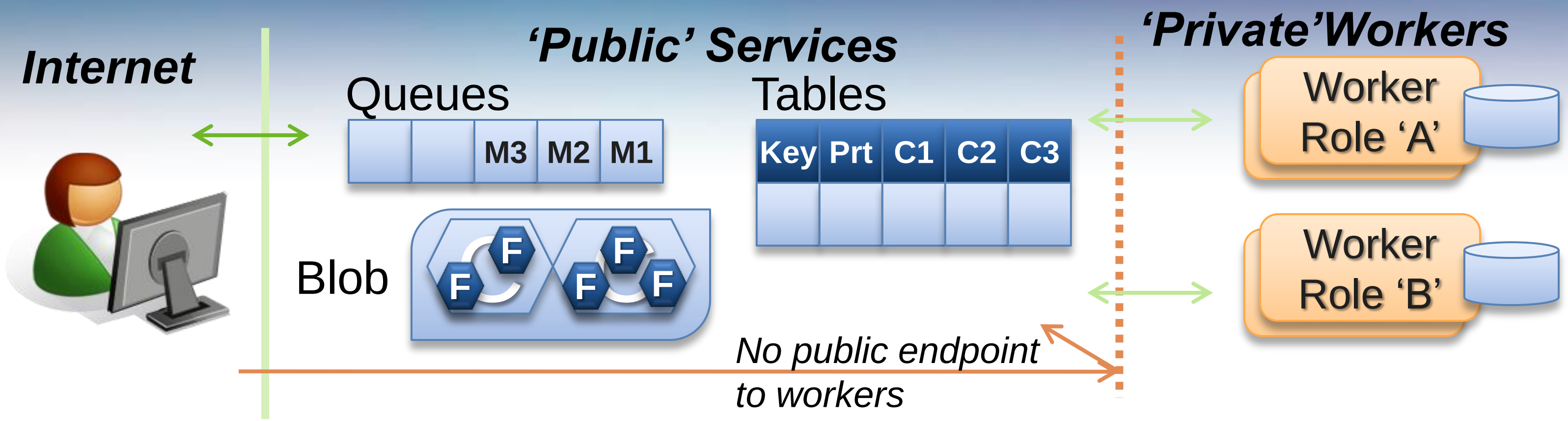
- Infrastructure as a Service (***IaaS***)
  - Virtualizes hardware as scalable services
- Platform as a Service (***PaaS***)
  - Provides scalable development platforms
- Software as a Service (***SaaS***)
  - Customize scalable applications

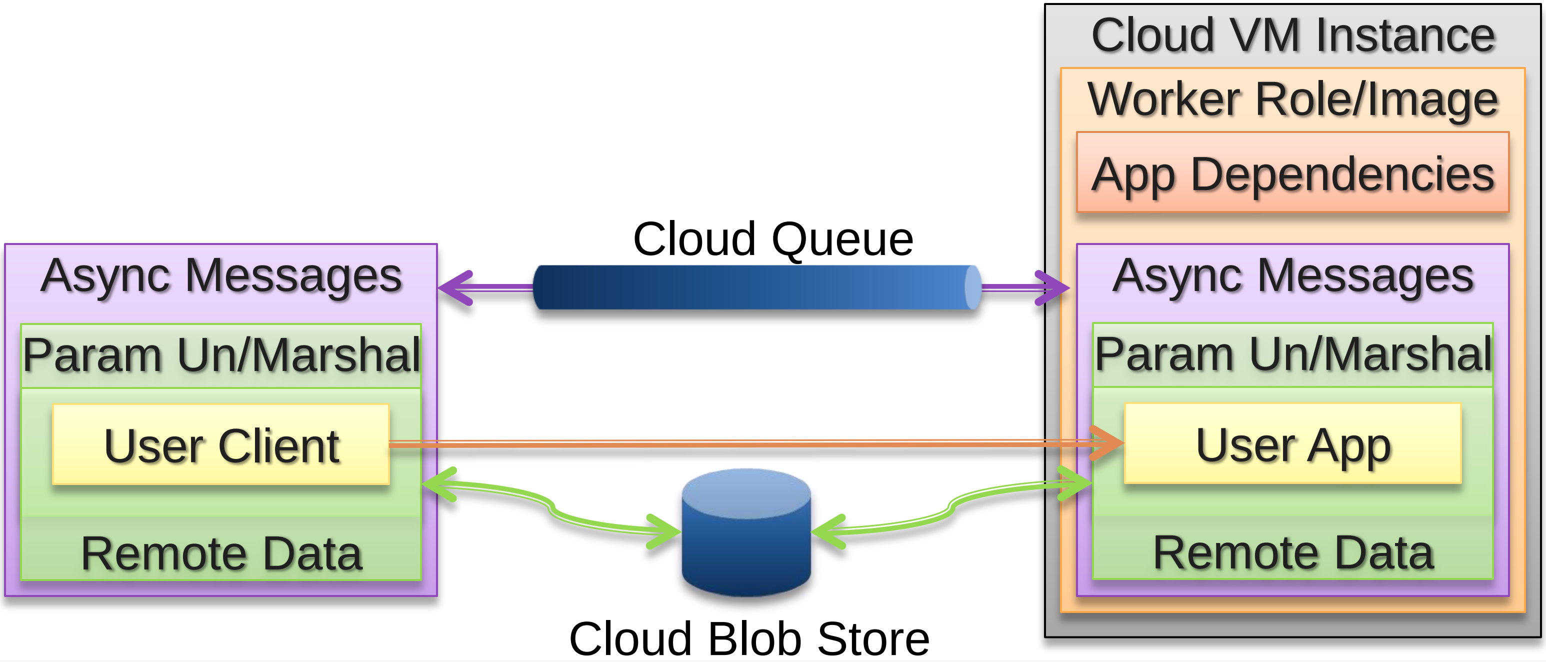


Challenges for Science Apps across Desktop & Cloud

- Migrating desktop apps to cloud, in part or full
- **Application Deployment**
  - Manage VM images, roles, dependencies
  - Redeploy upon any change
- **Application Execution**
  - Rewrite apps & clients for async remote exec.
  - Param passing, queues, service wrappers
- **Data Access**
  - Access to local files by client & cloud apps

Migrating Apps to the Azure Cloud





Goal

Improve accessibility to Cloud:

- **Deploy** Apps from desktop to cloud
- **Invoke** Cloud Apps from desktop clients and workflows
- Support efficient and automated **file access** across desktop and cloud

with minimal user overhead and code change

» *Generic Worker Framework for Azure*

Common worker role capable of dynamic deployment & execution of registered application from simple desktop commandline clients, workflows or APIs

Application Deployment
Register App with Generic Worker

- ***Register*** application to make them available for on-demand deployment
- ***Self-contained*** .NET, .exe, Java apps
- Commandline tool: ***register.exe***
- Pass location of ***bin*** directory with app files and unique dependencies
- Pass "***runtime type***" of application for shared dependencies
- Specify ***method*** to run or .exe signature

Application Deployment
Sample App Register Command

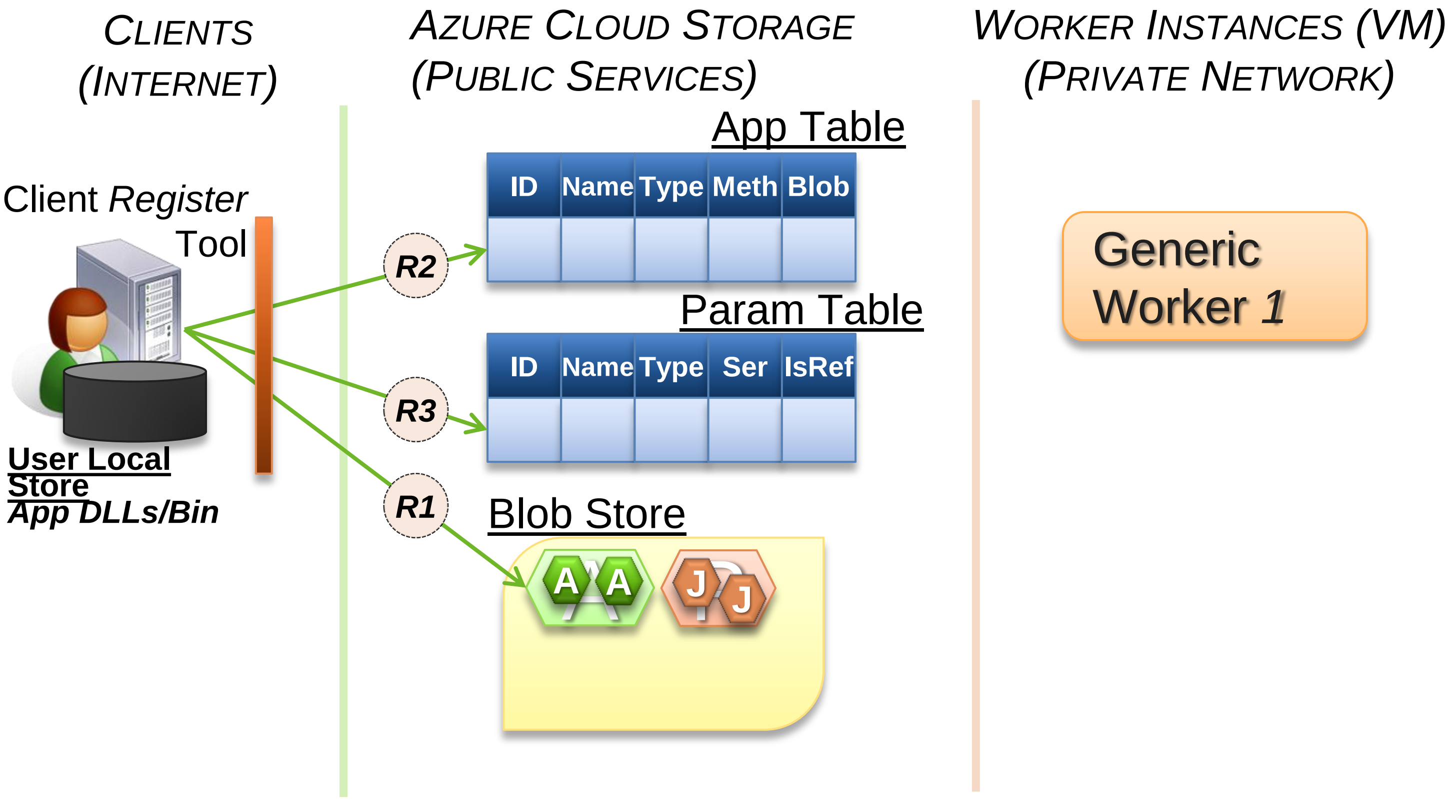
(a)

```
class MathOps {
  int Add(int i, int j) { return i + j; }
  int Mult(int i, int j) { return i * j; }
}
```

(b)

```
> register -type .net35 -name MyMathOps
    -class SampleCloudApp.MathOps
    –appDir c:\SampleCloudApp\bin
> register -type bin -name MyBlastAll
    -cmd “blastall –p blastn –d refseq_rna
-i {1} -o {2}”
    -in #1 file  #2 string  -out #2 file
    –appDir c:\ncbi\blast\bin
```

Application Deployment
App Register Operations



CLIENTS (INTERNET)

AZURE CLOUD STORAGE (PUBLIC SERVICES)

WORKER INSTANCES (VM) (PRIVATE NETWORK)

App Table

ID	Name	Type	Meth	Blob

Param Table

ID	Name	Type	Ser	IsRef

Blob Store

Generic Worker *1*

Application Execution

- Allow desktop clients to run cloud apps
  - Commandline tool for invocation: *invoke.exe*
  - Pass app input parameters on commanline
  - Internal *Invoker* .NET library to marshal params into XML message, Put on Job Request queue, Poll for response
  - Output XML Message unmarshalled as .NET objects
- Multiple apps can run concurrently in a single worker

Application Execution
Sample App Invoke Command

(a)

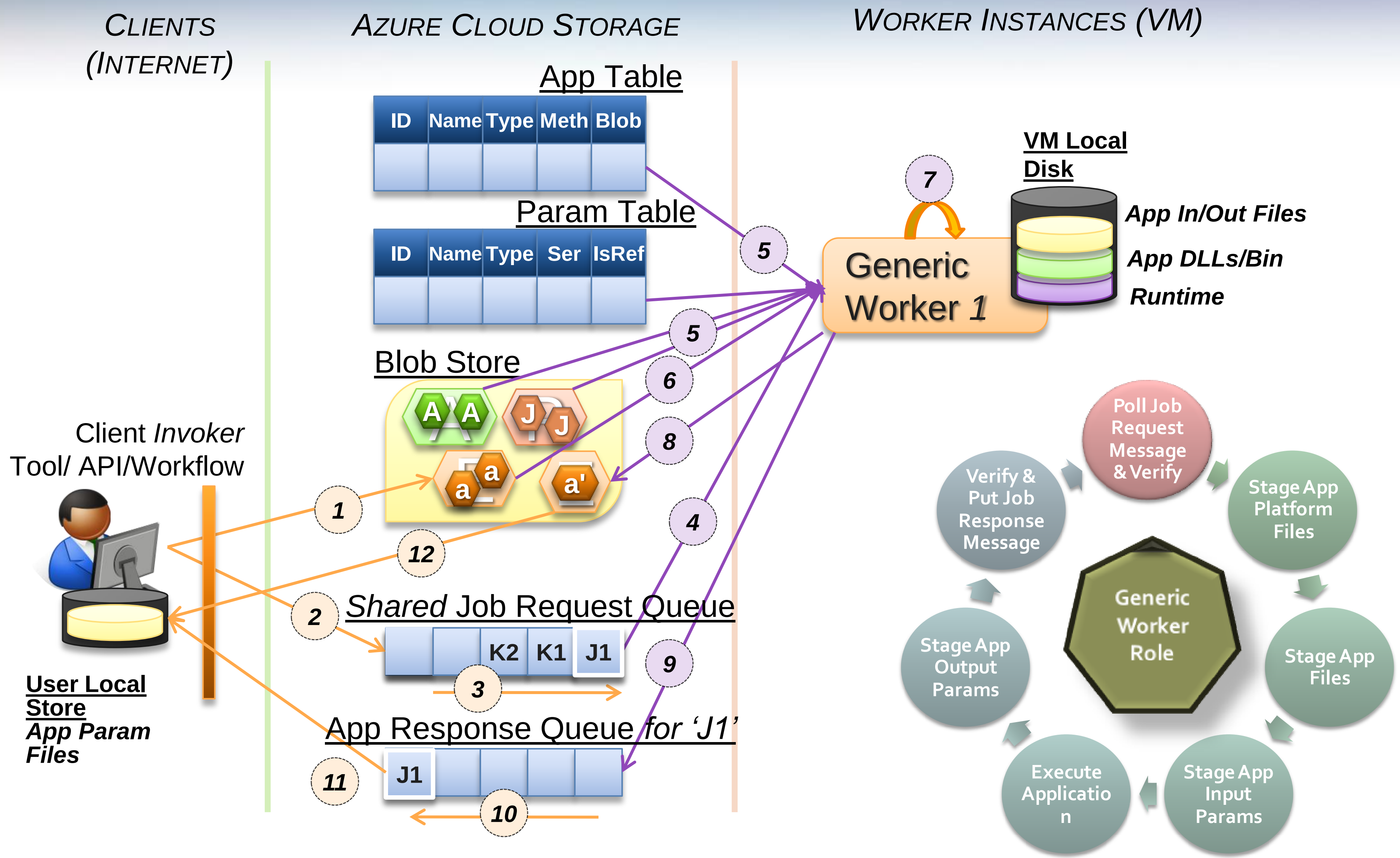
```
> invoke MyMathOps.Add 1 5
Return value: 6
> invoke MyBlastAll input.fasta output.txt
Return value: c:\workdir-036\output.txt
Download Console.Out file (y/n)?
```

(b)

```
// int s = (new MathOps()).Add(1,5);
Invoker invkr = new Invoker(“MyMathOps);
int s = (int)invkr.Invoke(“Add",new[]{1,5});
```

Application Execution

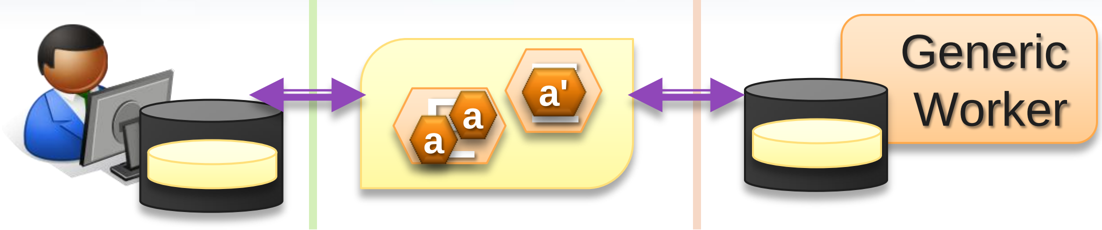
App Invoke Operations



Data Access

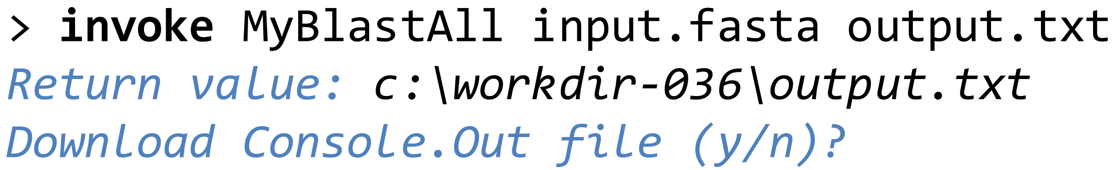
Automatic & Efficient Data Transfers

- Desktop:Worker file transfer thru Blob Store



- Generic Worker recognizes “file” params during registration
- Framework does automatic JIT data transfer
  - Clients & apps continue to pass local file paths
- File “references” ensure transfer if needed
- Basic caching to reduce cloud:local transfers

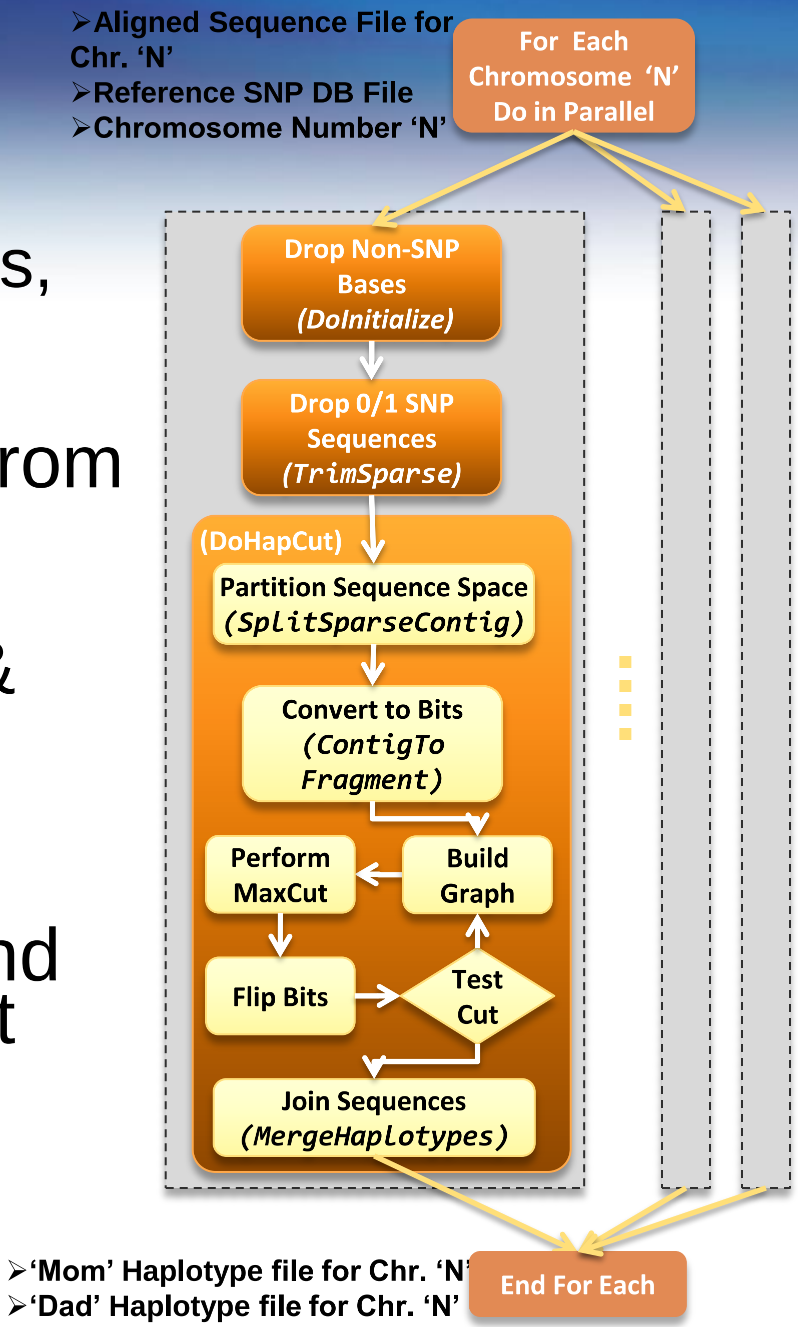
(a)



```
> invoke MyBlastAll input.fasta output.txt
Return value: c:\workdir-036\output.txt
Download Console.Out file (y/n)?
```

Genome Phasing

- Microsoft Biology Foundation
  - Library of genomics algorithms, data structures & file parsers
- Genome phasing separates parent haplotype sequences from sequence reads
- Phasing workflow includes compute intensive algorithm & pre-, post- procesors
- Achieve parallelism across chromosomes
- Goal: Orchestrate workflow and pre-, post- activities locally but ship compute heavy to cloud



➢Aligned Sequence File for Chr. ‘N’
➢Reference SNP DB File
➢Chromosome Number ‘N’

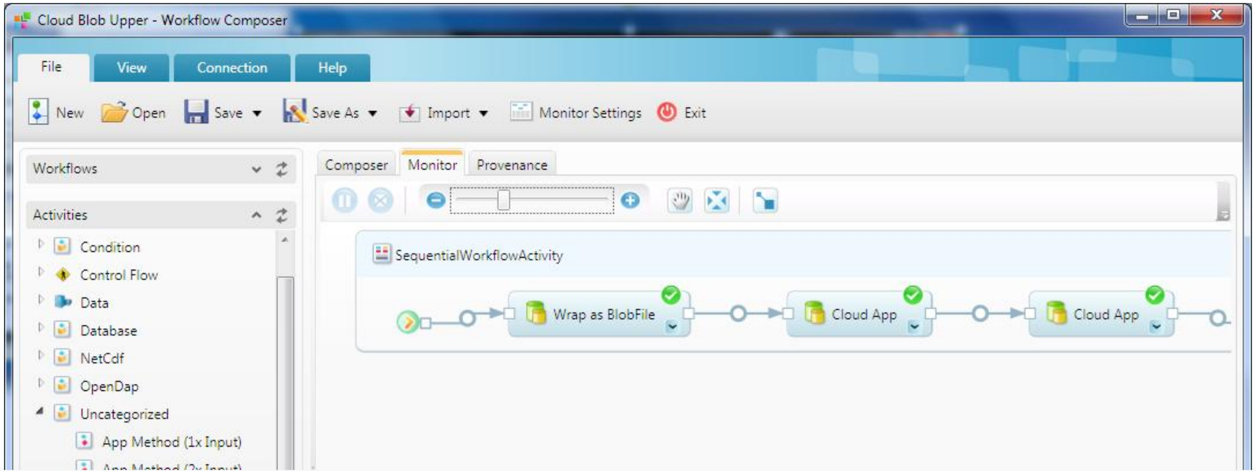
For Each Chromosome ‘N’ Do in Parallel
Drop Non-SNP Bases (DoInitialize)
Drop 0/1 SNP Sequences (TrimSparse)
(DoHapCut)
Partition Sequence Space (SplitSparseContig)
Convert to Bits (ContigTo Fragment)
Build Graph
Perform MaxCut
Flip Bits
Test Cut
Join Sequences (MergeHaplotypes)
End For Each

➢‘Mom’ Haplotype file for Chr. ‘N’
➢‘Dad’ Haplotype file for Chr. ‘N’

With David Heckerman, Simon Mercer & Michael Zyskowski

Genome Phasing Workflow

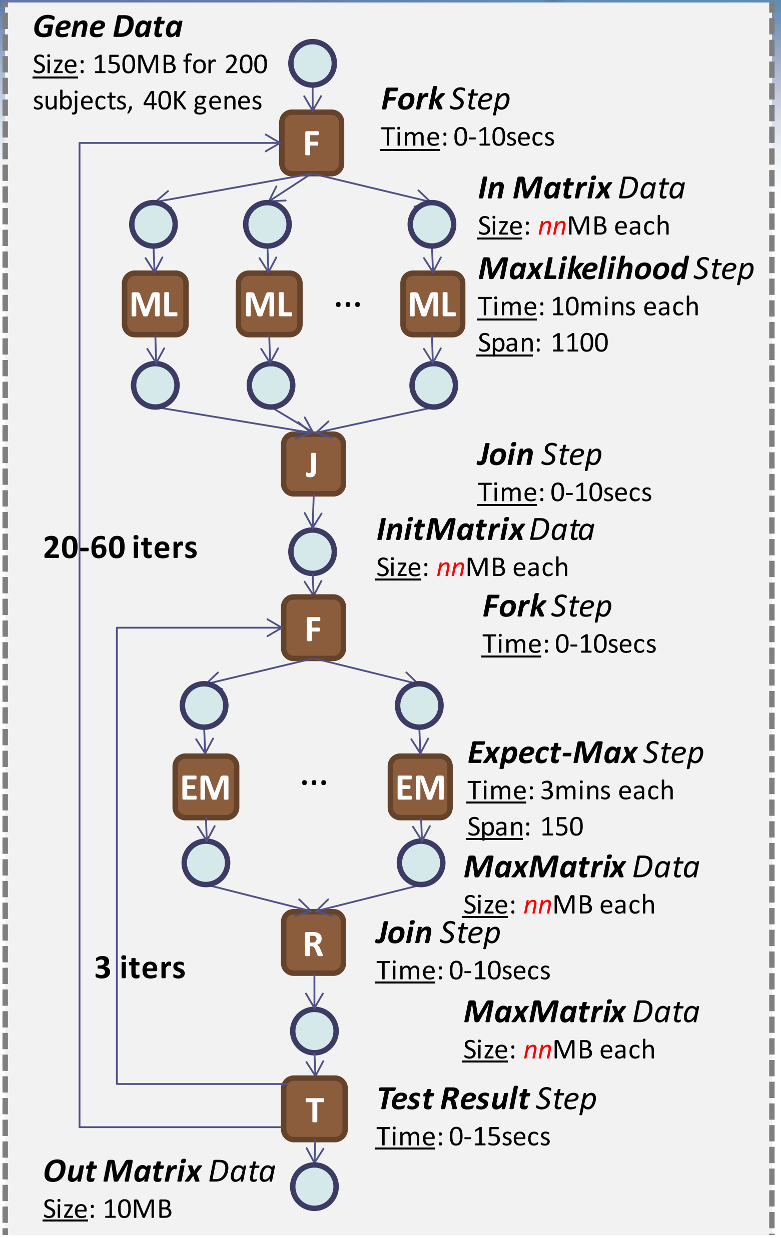
- Trident as workflow environment
- *CloudApp* activity can invoke any registered app in Generic Worker using API
- Data transfers are transparent
- Potential to host a suite of composable MBF genomics algorithms in Cloud



Genome Wide Association Study

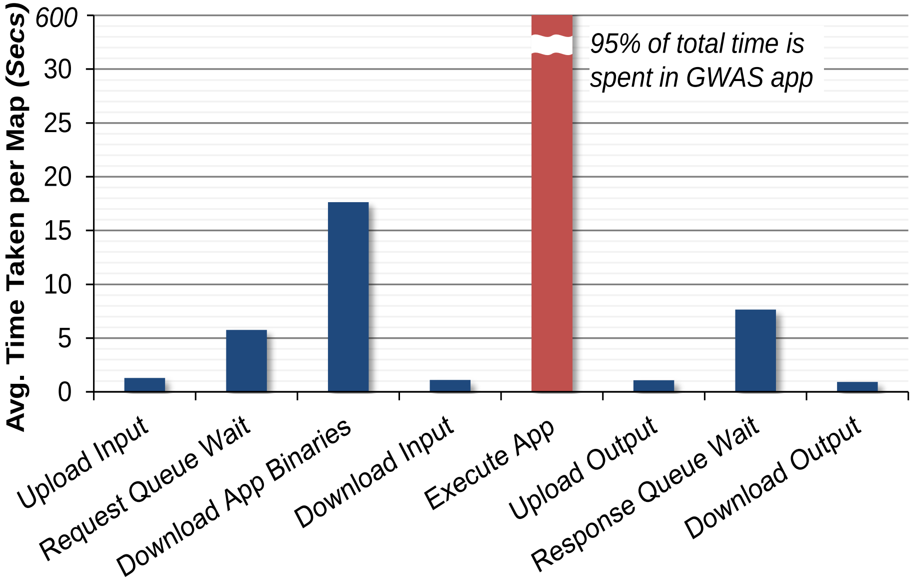
- Iterative MapReduce of ML & EM methods
- Compute intensive
  - ~18hrs on 300 HPC cores
- .NET code orchestrates execution, batch submission, file & parameter passing
- Need to share with research community

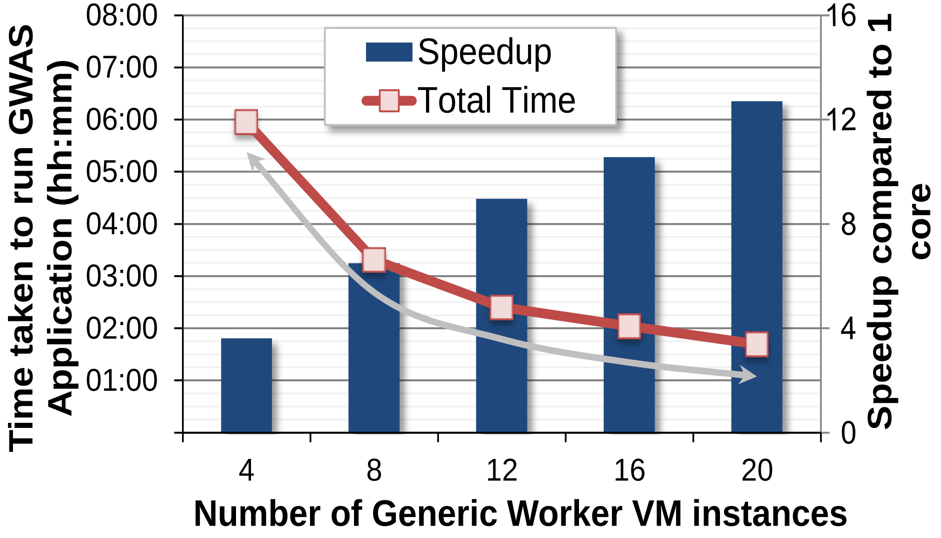
With Jennifer Listgarten & David Heckerman



GWAS application on Azure

- ML and EM methods registered as apps
- Fork-Join to Generic Workers replaced HPC parametric sweep batch submission





- Allow easy switching between HPC and Cloud environments

Future Work

- Support further application runtime types
  - Beyond .NET & Exe to Java, MatLab, etc.
- Dynamic worker instance scale-out based on collective demand
- Leverage data locality in worker file cache
  - E.g. workflow data pipelines
- Metrics for automated cross-scheduling workflows across Desktop, HPC and Cloud

Conclusions

- Cloud has opportunities to democratize science, but has to be accessible
- Reducing overhead to migrating existing desktop apps to Cloud can drive adoption
- App deployment, remote execution and file handling often require code change
- Generic Worker reduces user and coding overhead with limited performance penalty
- Intuitive execution across desktop & Cloud from commandline, workflows & APIs