Processing next-gen sequencing data through a Pegasus workflow

2011-9-19

Yu Huang

Brief Introduction to Vervet Project

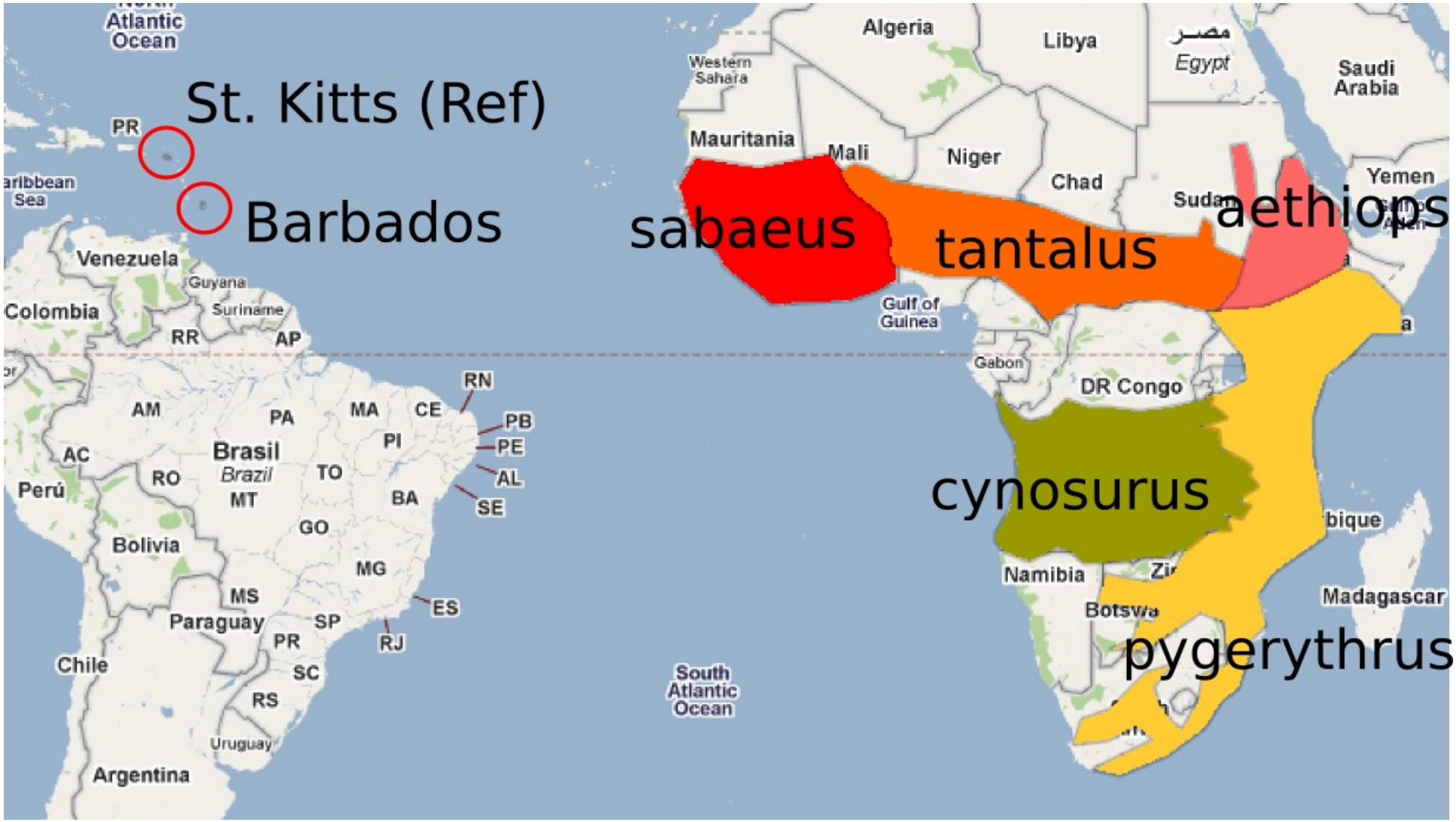
Who are vervets (African green monkeys)?

- Old world monkey from sub-Saharan Africa
- ~16-18 MYA (Millions of Years Ago): MRCA of vervet and macaque. (v.s. MRCA of macaque and human ~26MYA.)
- ~1.4-5 MYA (fossil + a few genes): Several vervet subspecies started to emerge. (v.s. MRCA of human and chimp ~6MYA)
- 1600s-1700s: A “small” number of vervets were brought to a few Carribean islands presumably through slave trade.

How is UCLA related to vervets?

- 1970s: Colony (VRC) with ~50 St. Kitts monkeys started at UCLA → Neurobehavioral research.
- 2000-2009: brain expression QTLs, microsatellite maps, etc.
- 2011: Reference is being assembled. ~110 Monkeys from VRC and 10 from St. Kitts have been sequenced.
- Late 2011: ~500 VRC monkeys.
- 2012: ~100s from Africa.

Geographic distribution



Main Questions

- Relationship between Barbados and the reference (St. Kitts)?
- Which African subspecies is the most likely source of Carribean monkeys?
- What's the relationship among African subspecies?
- Polymorphism level within VRC

Why do we care about questions aforementioned?

Genetic & phenotypic diversity level is key to trait mapping.

History is free genetic experiments.

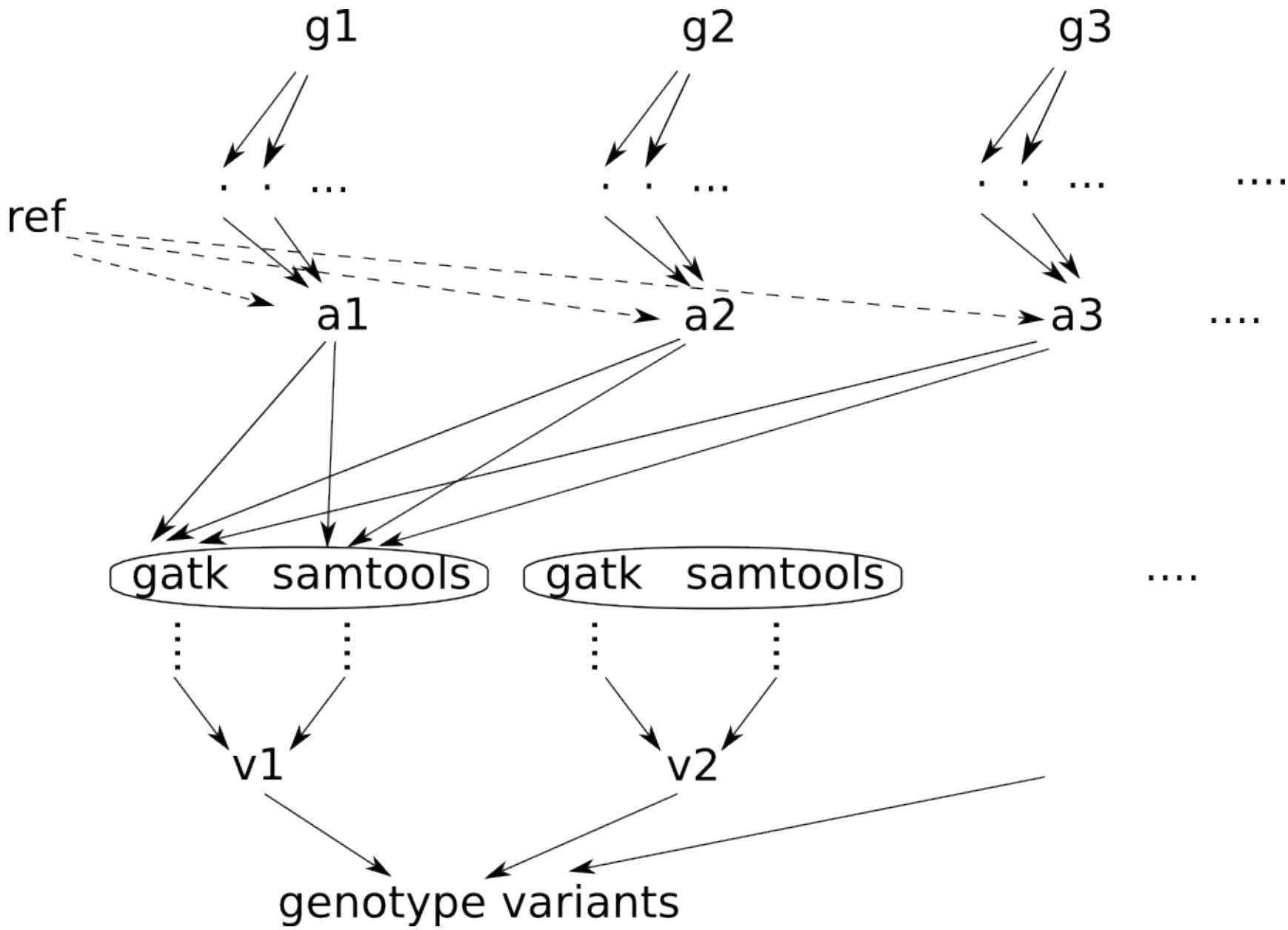
Geographic distribution mimics different stages of history happening simultaneously.

But why do we choose vervets to spend $$$?

- Widely available for research (v.s. Rhesus).
- Model to study high-level phenotypes: novelty-seeking, bipolar disorder.
- Model to study primate-specific diseases: SIV-infection, etc.
- Human curiosity.

The part relevant to ICNN

Pipeline: from short reads to genetic variants


g1
g2
g3
ref
a1
a2
a3
gatk samtools
gatk samtools
v1
v2
genotype variants

Usual Approach

Divide the whole pipeline into several steps.

Write scripts that submit qsub jobs for each step.

Run these submission scripts sequentially.

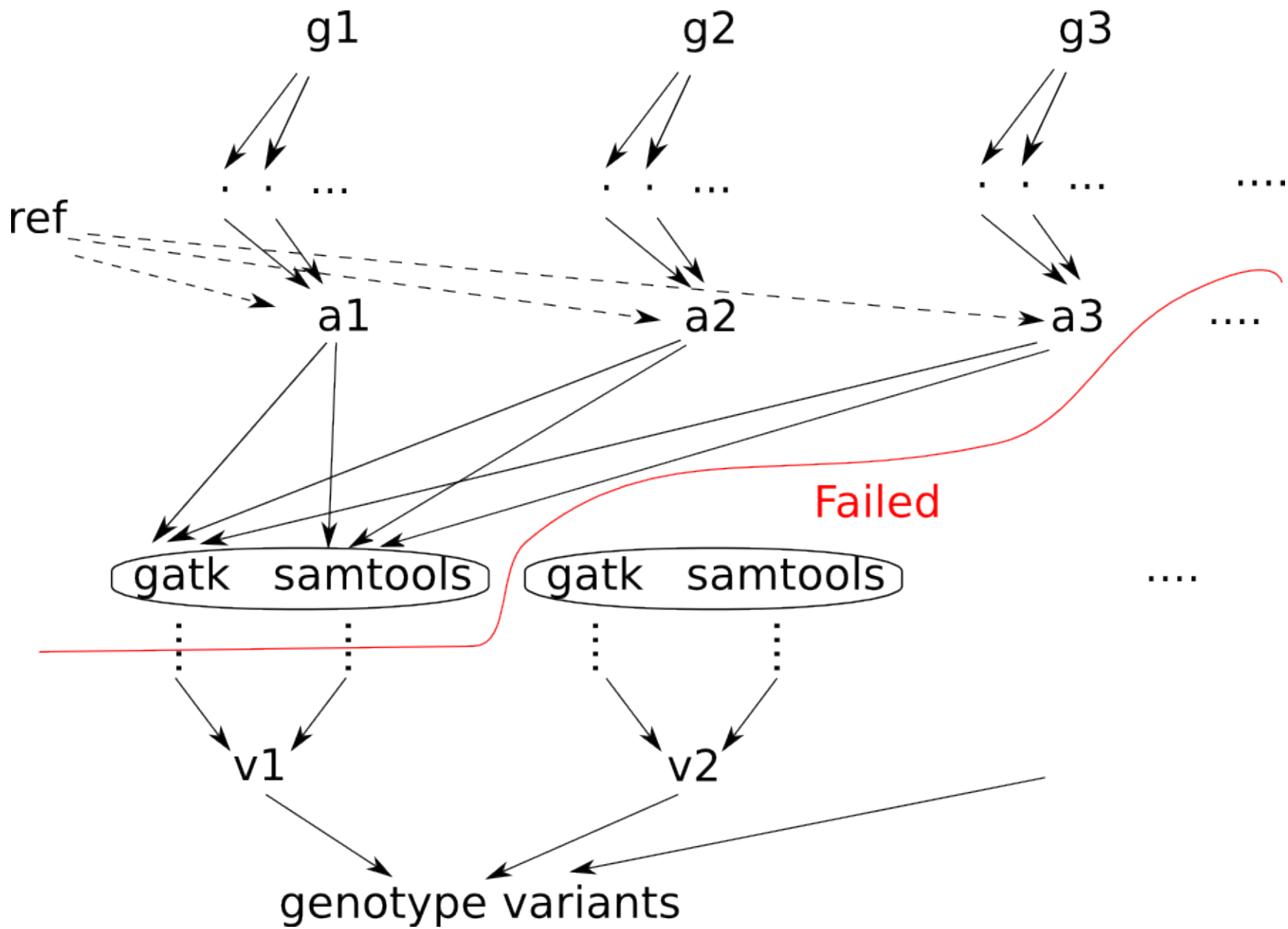
Sometimes, crank several steps into one shell script by using UNIX pipes. (Be careful about broken pipes.)

1. How do you know each job finishes properly.

Assuming the pipeline has been tested successfully,

- Program segmentation fault
- Computing machine failure
- Memory blowout, job killed
- Errors in intermediate UNIX pipes go undetected
- ...

2. How to resume a half-finished pipeline?



3. How to handle non-linear dependency?

- Add a checking script to make sure dependent jobs have proper output.
- Or run all these sequentially in one shell script (no parallel).

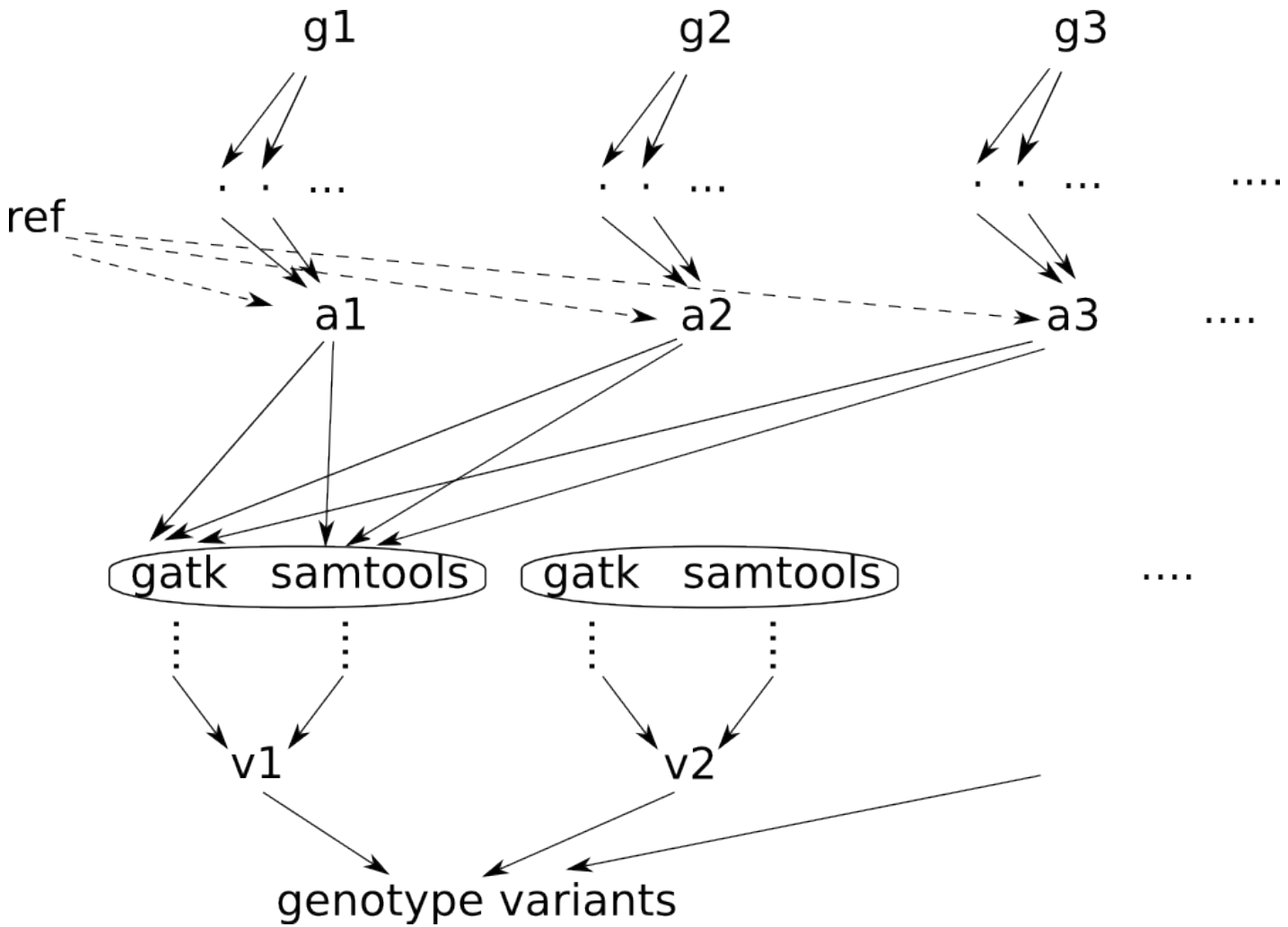
How does Pegasus find out failed jobs?

```
crocea@dl324b-1:/Network/Data/vervet/vervetPipeline$ pegasus-status -l
    ./work/AlignmentToCallPipeline_552_554_557_605_615_vs_524_top_156Contigs_uschpc_bugfix/

AlignmentToCallPipeline-0.dag is running.

09/17/11 12:48:25  Done     Pre   Queued    Post   Ready   Un-Ready   Failed

09/17/11 12:48:25   ===     ===      ===     ===     ===        ===      ===

09/17/11 12:48:25  6427       0        3       0       0       1392        4


WORKFLOW STATUS : Running | 6427/7826 ( 82% ) | (condor processing workflow)

crocea@dl324b-1:/Network/Data/vervet/vervetPipeline$ pegasus-analyzer -d
    ./work/AlignmentToCallPipeline_552_554_557_605_615_vs_524_top_156Contigs_uschpc_bugfix/

...
```

How to resume a half-finished pipeline?

```
crocea@crocea:/Network/Data/vervet/vervetPipeline$ ./reSubmit.sh
    work/AlignmentToCallPipeline_552_554_557_605_615_vs_524_top_156Contigs_uschpc_bugfix/
AlignmentToCallPipeline-0.dag is running.
09/17/11 12:48:25  Done     Pre   Queued    Post   Ready   Un-Ready   Failed
09/17/11 12:48:25   ===     ===      ===     ===     ===        ===      ===
09/17/11 12:48:25  6427       0        7       0       0       1392        0

WORKFLOW STATUS : Running | 6427/7826 ( 82% ) | (condor processing workflow)
```

Non-linear dependency? => DAG (Directed Acyclic Graph)



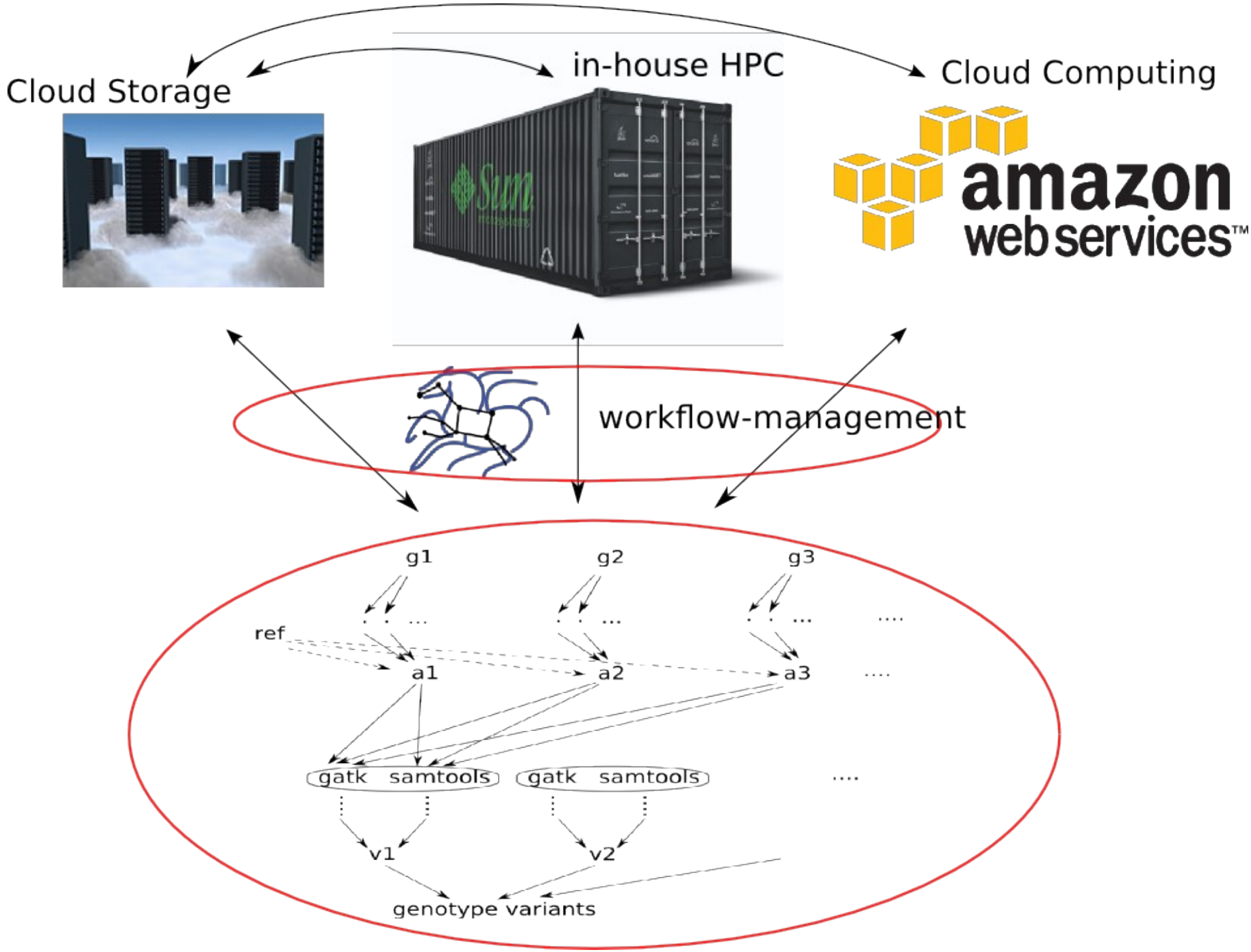
What is Pegasus? pegasus.isi.edu

Pegasus is a configurable system for mapping and executing abstract application workflows over a wide range of execution environment including a laptop, a campus cluster, a Grid, or a commercial or academic cloud.

Today, Pegasus runs workflows on Amazon EC2, Nimbus, Open Science Grid, the TeraGrid, and many campus clusters.

One workflow can run on a single system or across a heterogeneous set of resources.

Pegasus can run workflows ranging from just a few computational tasks up to 1 million.


Cloud Storage
in-house HPC
Cloud Computing
amazon
web services™
workflow-management
g1
g2
g3
ref
a1
a2
a3
gatk
samtools
gatk
samtools
v1
v2
genotype variants

Summary of advantages over usual approach

- Check failed jobs
- Restart half-finished pipelines
- Complicated dependency
- Conditional dependency (if something is not there, add a job to generate it)

Other advantages

- Multi-cluster capability (don't need to deal with cluster diversity).
- Automatic re-trial on failed jobs for 3 times (error due to creation of the same directory, node failure/reboot, disk quota, memory blowout).
- Utilize the cluster resource more efficiently. Start whenever the parent job is finished.
- Saves lots of time as you skip qsub scripts, cleanup scripts, data integrity checking scripts.
- Combine multiple workflows.

Analyze your workflow

- Which step is failing most?
- Which step uses most time?

Transformation	Count	Mean	Variance	Min	Max	Total
pegasus::dirmanager	1	0.1860	0.0000	0.1860	0.1860	0.1860
workflow::vcf_convert:1.0	1194	14.2852	685.3181	0.1120	404.4430	17056.5190
workflow::vcf_isec:1.0	597	1.1518	0.2919	0.1010	5.0990	687.6070
workflow::mkdirWrap:1.0	5	0.0394	0.0002	0.0240	0.0610	0.1970
pegasus::cleanup	3561	0.2605	0.0717	0.1030	6.1300	927.7870
pegasus::pegasus-transfer	13	75.3040	9843.8631	0.3880	341.0510	978.9520
workflow::vcf_concat:1.0	156	1.1243	0.2526	0.4320	4.2660	175.3840
workflow::bgzip_tabix:1.0	1194	0.5487	0.0554	0.0760	3.5100	655.1820
workflow::genotyperJava:1.0	599	293.9758	20062.3637	0.0810	869.7040	176091.4820
workflow::CallVariantBySamtools:1.0	597	157.3317	6341.1175	3.7520	811.4270	93927.0150

#Transformation - name of the transformation
#Count - the number of times the transformation was executed
#Mean(sec.) - the mean of the transformation runtime
#Variance(sec.) - the variance of the transformation runtime
#Min(sec.) - the minimum transformation runtime value
#Max(sec.) - the maximum transformation runtime value
#Total(sec.) - the cumulative of transformation runtime

Advantage over embarrassingly simple parallel MPI/Hadoop

- Easy to debug. Debug/Test program individually. No crazy “printf” debugging for mpi programs.
- Resistant to partial node/job failure, retry the job

  elsewhere

Advantage over LONI&Kepler

Scale

- Can you generate a workflow of 10s of thousands of jobs through mouse clicking?
- LONI&Kepler seem to be designed for non-programmers.

Downside of Pegasus

- Architecture is a bit complicated pegasus -> condor_g -> globus -> SGE
  - Although a simpler setup, pegasus -> condor, is available if you have administration rights over the cluster.
  - globus is the weak link in the complicated setup.
- It spends about ~60s in managing each job. Bad if you have lots of short jobs. (job clustering could mitigate this).
- Data passing between jobs is through files, rather than direct memory/network transfer in MPI/Hadoop. Unix pipe could mitigate this.

Demo Run