Assignment 3 to 8 – Project Themes

Supercomputers - Cyberinfrastructure

Anatomy of a Science Gateway

- Gateway User Interface
  - Web Portals
  - Desktop Clients
  - Social/ Collaboration Capabilities
- Security Infrastructure
- Analyses & Visualization Capabilities
- Execution Frameworks
  - Application Abstraction
  - Workflow construction & Enactment
  - Compute Resource Management
  - Scheduling
  - Messaging System
- Data Management
- Provenance Collection

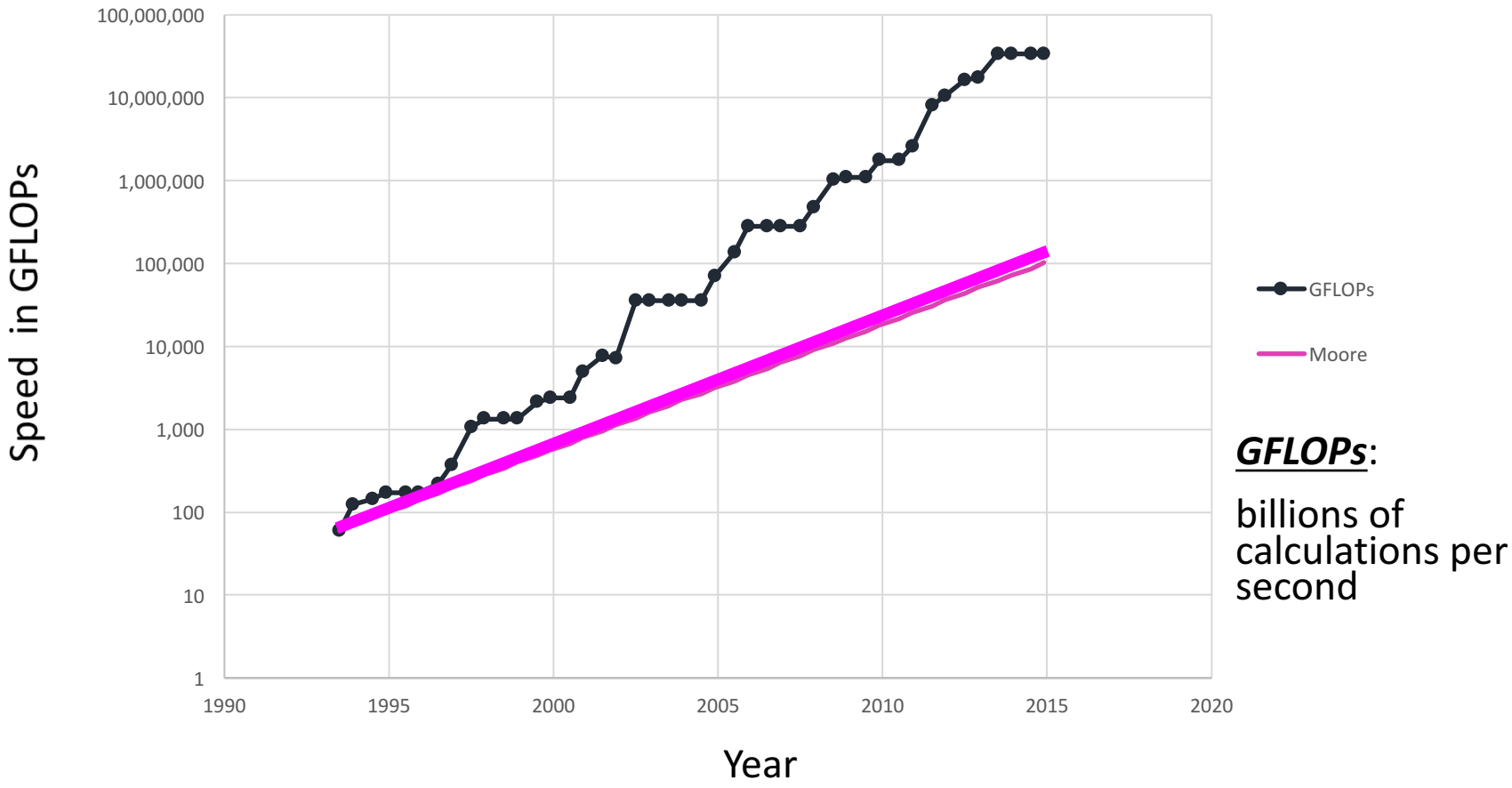
Supercomputer?

What is Supercomputing?

- ***Supercomputing*** is the **biggest, fastest computing right this minute**.
  - So, the definition of supercomputing is **constantly changing**.
  - GigaFLOPS, TeraFLOPS, PetaFLOPS, ExaFLOPS……
- Supercomputing is also heard in context of :
  - *High Performance Computing* (HPC),
  - *High End Computing* (HEC),
  - Grid Computing
  - *Cyberinfrastructure* (CI).

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Fastest Supercomputer vs. Moore



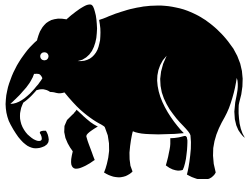
GFLOPs:

billions of calculations per second

Top500.org: Listing of 500 most powerful computers in the world

What is Supercomputing About?

- **Size**: Many problems that are interesting to scientists and engineers **can't fit on a PC** – usually because they need more than a few GB of RAM, or more than a few 100 GB of disk.

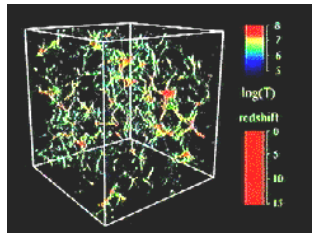


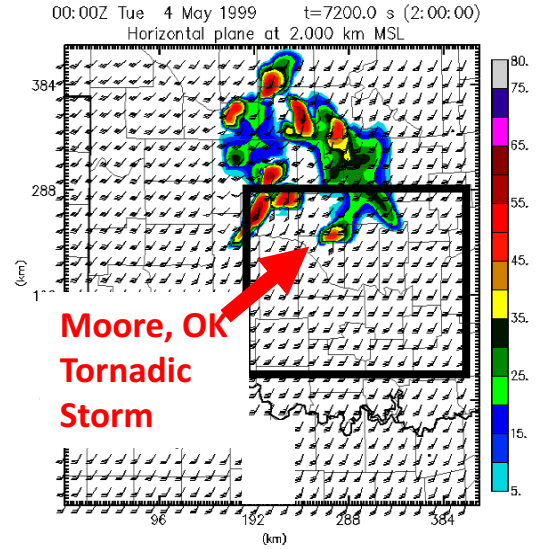
- **Speed**: Many problems that are interesting to scientists and engineers would take a very very long time to run on a PC: months or even years. But a problem that would take **a month on a PC** might take only **an hour on a supercomputer**.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

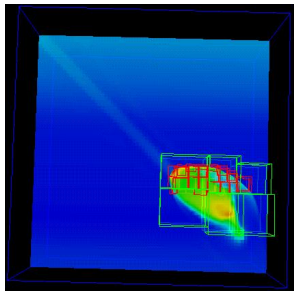
What is HPC Used For?

- ***Simulation*** of physical phenomena, such as
  - Weather forecasting
  - Galaxy formation
  - Oil reservoir management





- ***Data mining***: finding **needles** of information in a haystack of data, such as
  - Gene sequencing
  - Signal processing
  - Detecting storms that might produce tornados



- ***Visualization***: turning a vast sea of **data** into **pictures** that a scientist can understand

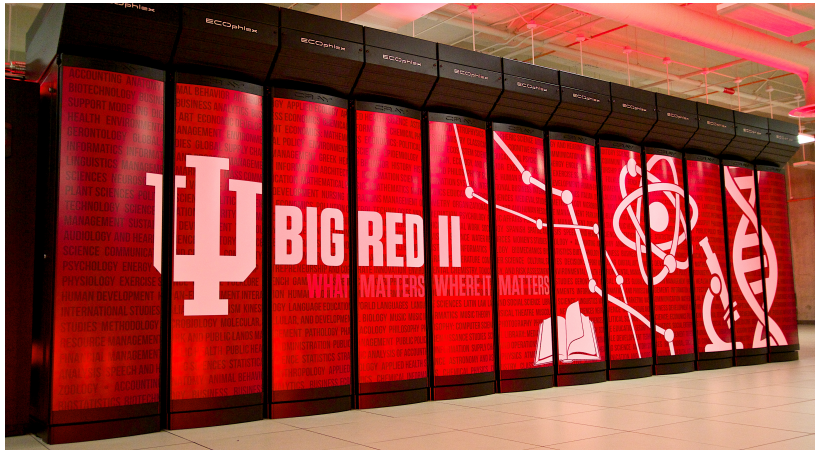
Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

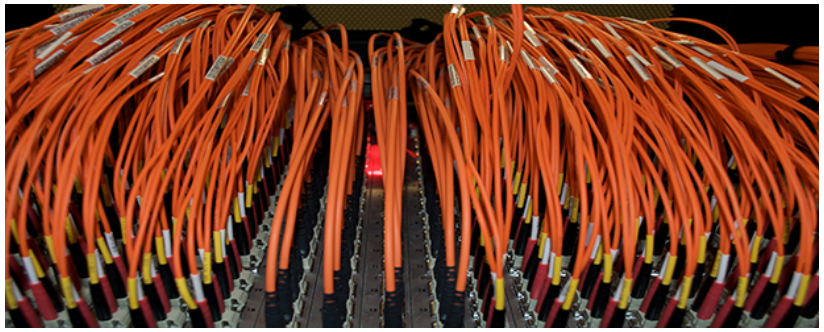
What a Cluster is ….

A cluster is a collection of small computers, called ***nodes***, hooked together by an ***interconnection network***.

It also **needs** software that allows the nodes to communicate over the interconnect.

A cluster **is** … is all of these components working together as if they're one big computer … a **super** computer.



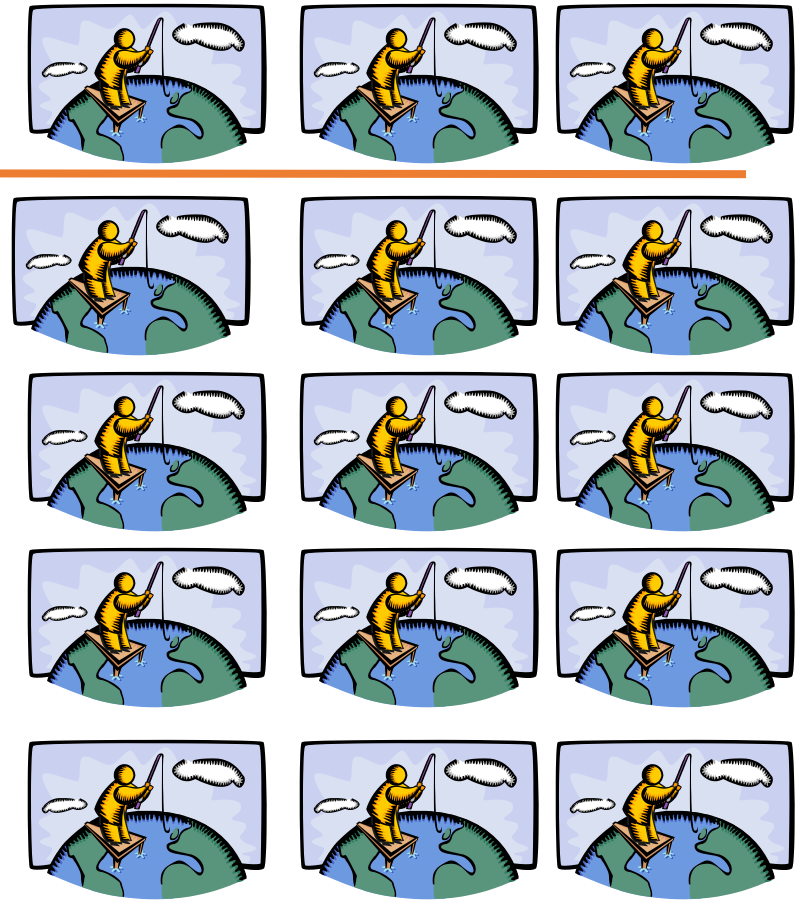


Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Parallelism

Parallelism means doing multiple things at the same time: you can get more work done in the same time.

Less fish …



More fish!

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

What is Parallelism?

Parallelism is the use of multiple processing units – either processors or parts of an individual processor – to solve a problem, and in particular the use of multiple processing units operating concurrently on different parts of a problem.

The different parts could be different tasks, or the same task on different pieces of the problem's data.

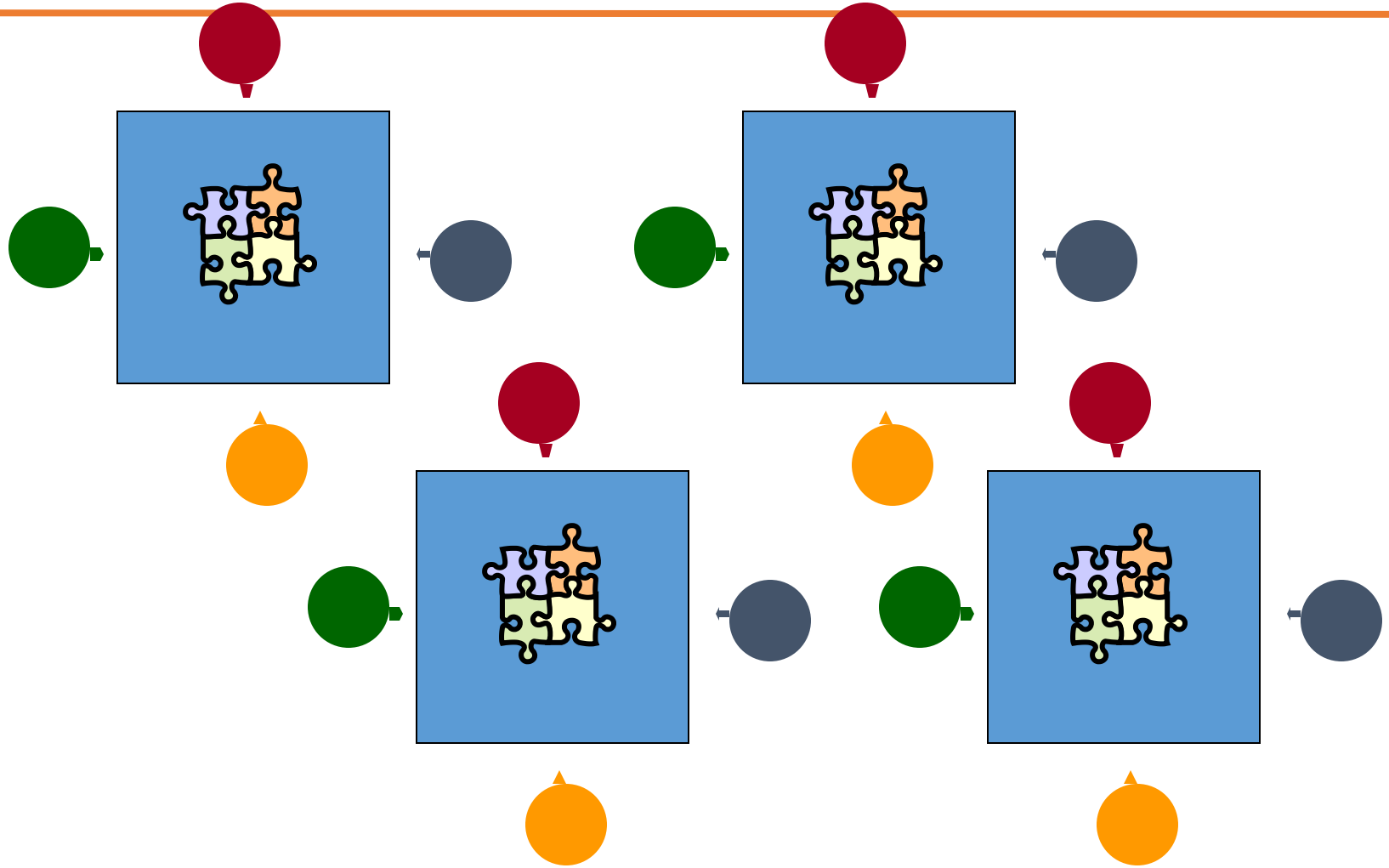
Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Kinds of Parallelism

- Instruction Level Parallelism
- Shared Memory Multithreading
- Distributed Memory Multiprocessing
- GPU Parallelism
- Hybrid Parallelism (Shared + Distributed + GPU)

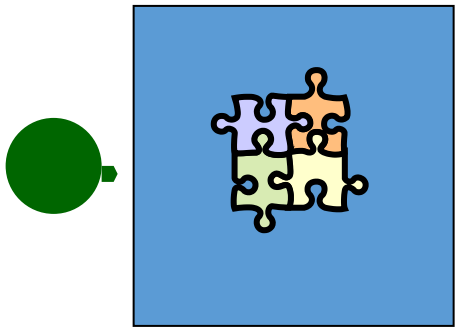
Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

The Jigsaw Puzzle Analogy



Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

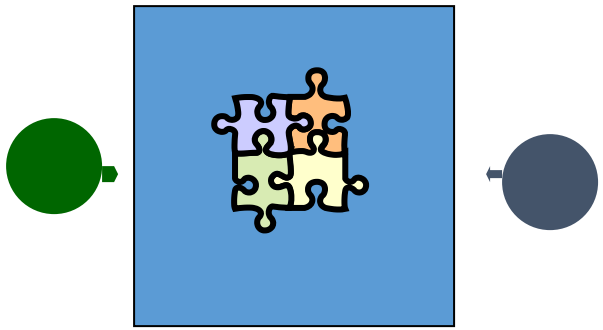
Serial Computing



- Suppose you want to do a jigsaw puzzle that has, say, a thousand pieces.
- We can imagine that it'll take you a certain amount of time.
- Let's say that you can put the puzzle together in an hour.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

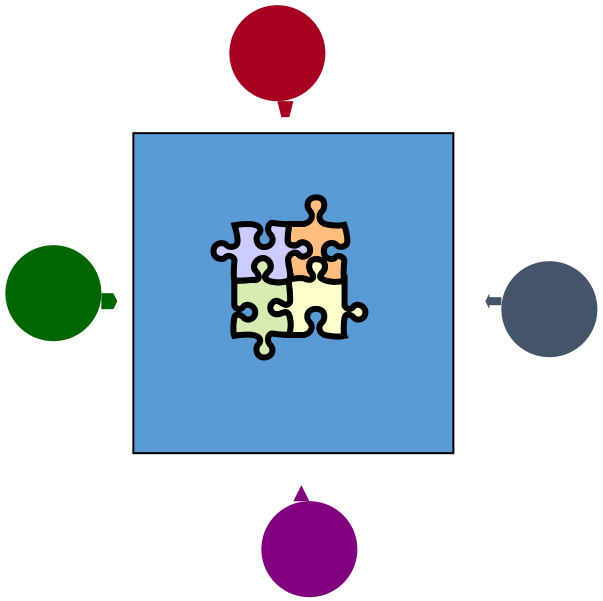
Shared Memory Parallelism



- If Scott sits across the table from you, then he can work on his half of the puzzle and you can work on yours.
- Once in a while, you'll both reach into the pile of pieces at the same time (you'll ***contend*** for the same resource), which will cause a little bit of slowdown.
- And from time to time you'll have to work together (***communicate***) at the interface between his half and yours.
- The speedup will be nearly 2-to-1: y'all might take 35 minutes instead of 30.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

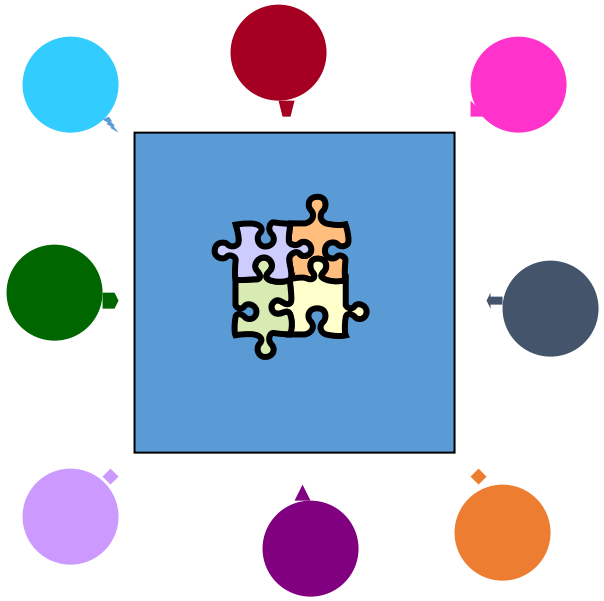
The More the Merrier?



- Now let's put Paul and Charlie on the other two sides of the table.
- Each of you can work on a part of the puzzle, but there'll be a lot more contention for the shared resource (the pile of puzzle pieces) and a lot more communication at the interfaces.
- So y'all will get noticeably less than a 4-to-1 speedup, but you'll still have an improvement, maybe something like 3-to-1: the four of you can get it done in 20 minutes instead of an hour.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Diminishing Returns



- If we now put Dave and Tom and Horst and Brandon on the corners of the table
- there's going to be a whole lot of contention for the shared resource, and a lot of communication at the many interfaces.
- So the speedup y'all get will be much less than we'd like; you'll be lucky to get 5-to-1.
- So we can see that adding more and more workers onto a shared resource is eventually going to have a diminishing return.

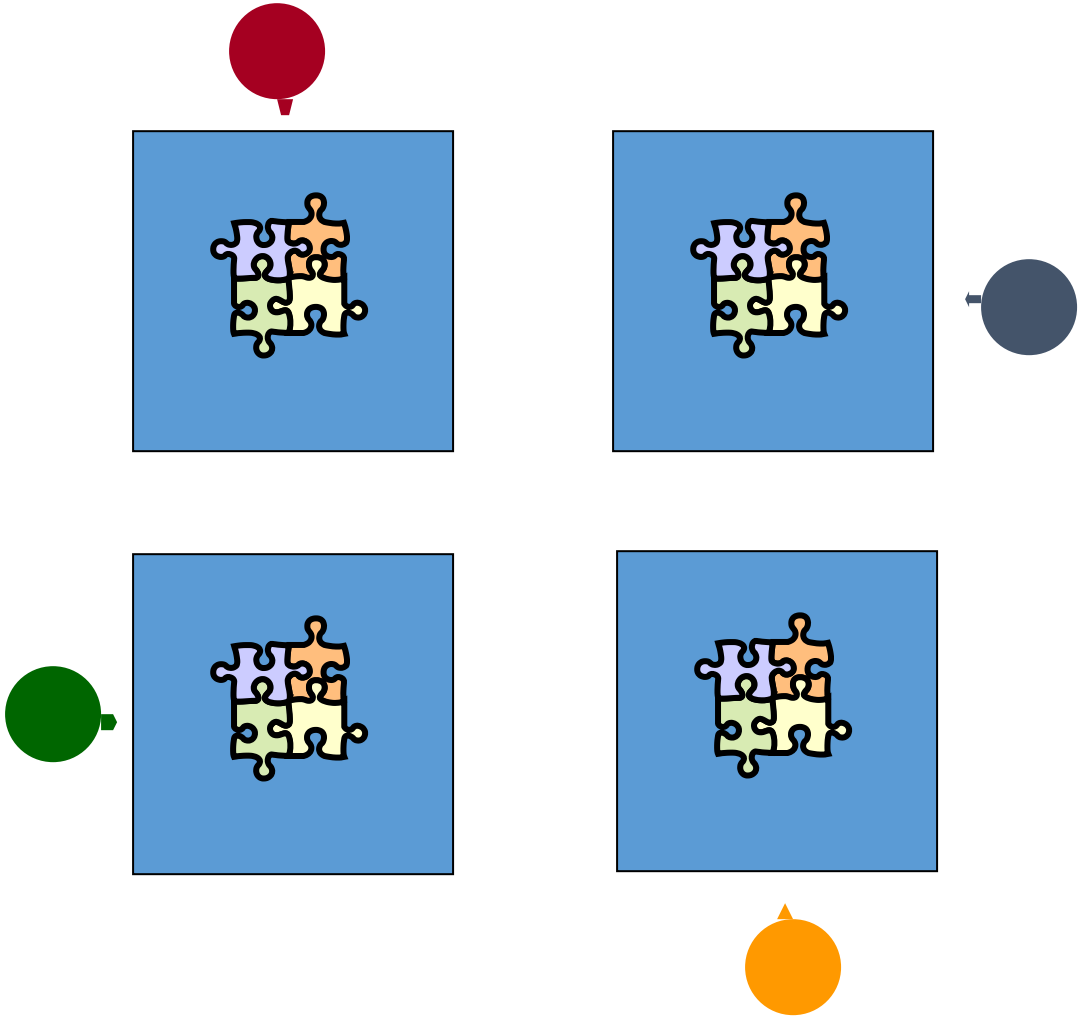
Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Distributed Parallelism

- Now let's try something a little different. Let's set up two tables, and let's put you at one of them and Scott at the other. Let's put half of the puzzle pieces on your table and the other half of the pieces on Scott's. Now y'all can work completely independently, without any contention for a shared resource. **BUT**, the cost per communication is **MUCH** higher (you have to scootch your tables together), and you need the ability to split up (***decompose***) the puzzle pieces reasonably evenly, which may be tricky to do for some puzzles.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

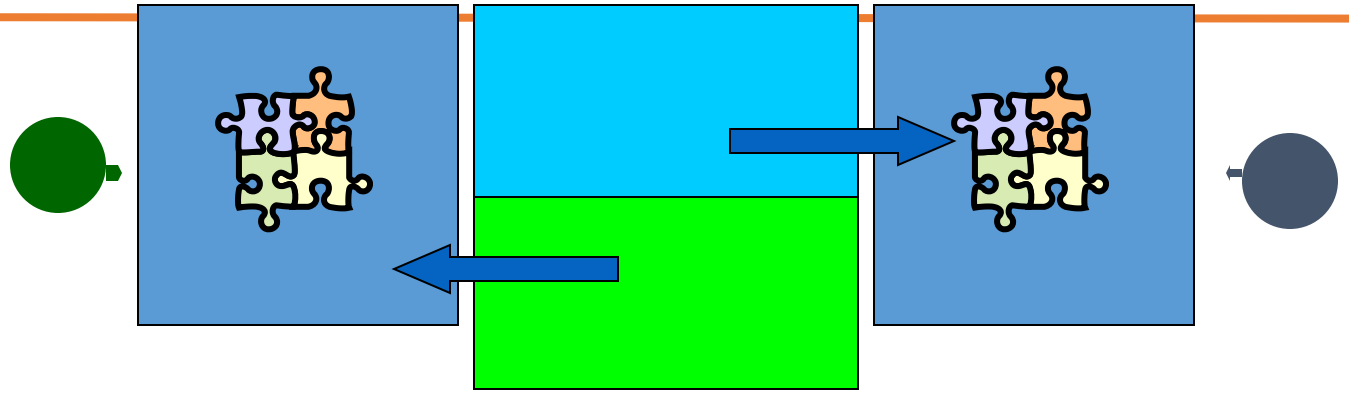
More Distributed Processors



- It's a lot easier to add more processors in distributed parallelism. But, you always have to be aware of the need to decompose the problem and to communicate among the processors.
- Also, as you add more processors, it may be harder to ***load balance*** the amount of work that each processor gets.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

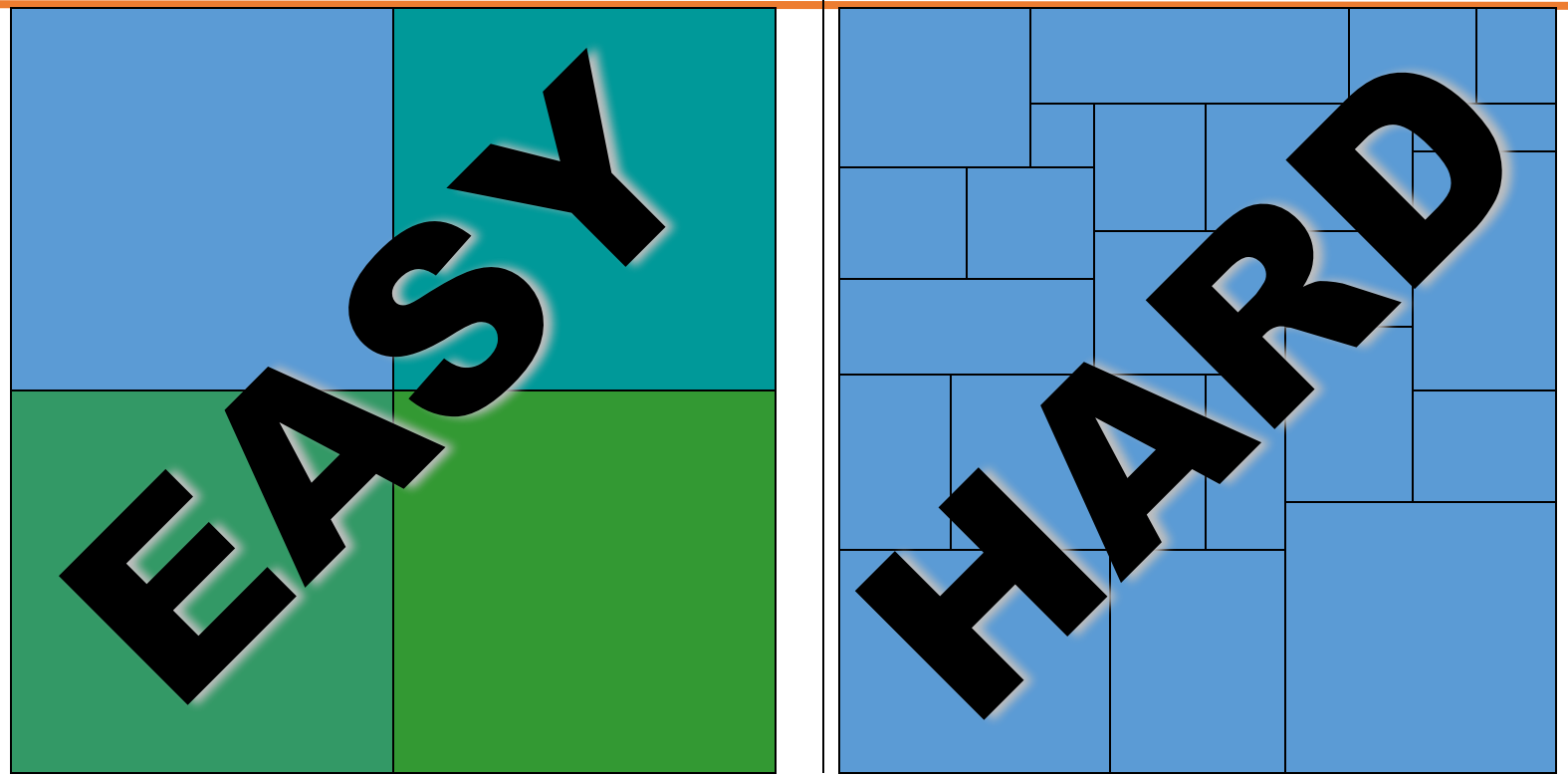
Load Balancing



- ***Load balancing*** means ensuring that everyone completes their workload at roughly the same time.
- For example, if the jigsaw puzzle is half grass and half sky, then you can do the grass and Scott can do the sky, and then y'all only have to communicate at the horizon – and the amount of work that each of you does on your own is roughly equal. So you'll get pretty good speedup.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Load Balancing



- Load balancing can be easy, if the problem splits up into chunks of roughly equal size, with one chunk per processor. Or load balancing can be very hard.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Why HPC is Worth the Bother

- What HPC gives you that you won't get elsewhere is the ability to do **bigger, better, more exciting science**. If your code can run faster, that means that you can tackle much bigger problems in the same amount of time that you used to need for smaller problems.
- HPC is important not only for its own sake, but also because what happens in HPC today will be on your desktop in about 10 to 15 years and on your cell phone in 25 years: it puts you **ahead of the curve**.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Gateways ***abstract*** these details from Users

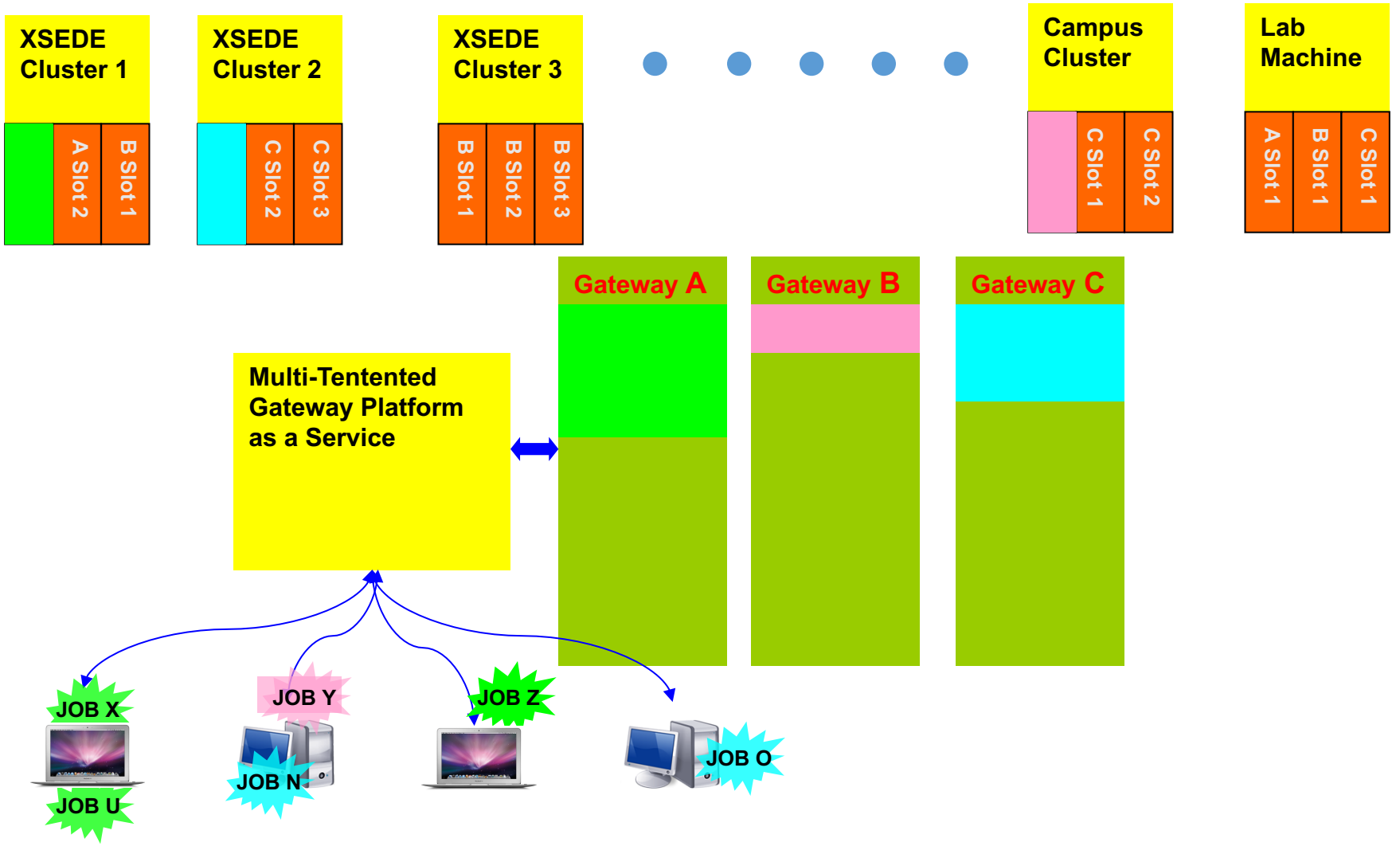
- Parallelism Types
  - Instruction Level Parallelism
  - Shared Memory Multithreading
  - Distributed Memory Multiprocessing
  - GPU Parallelism
  - Hybrid Parallelism (Shared + Distributed + GPU)
- Gateways pick the right computations paradigm based on the problem.

Source: Supercomputing in Plain English, Henry Neeman, University of Oklahoma

Science Gateways

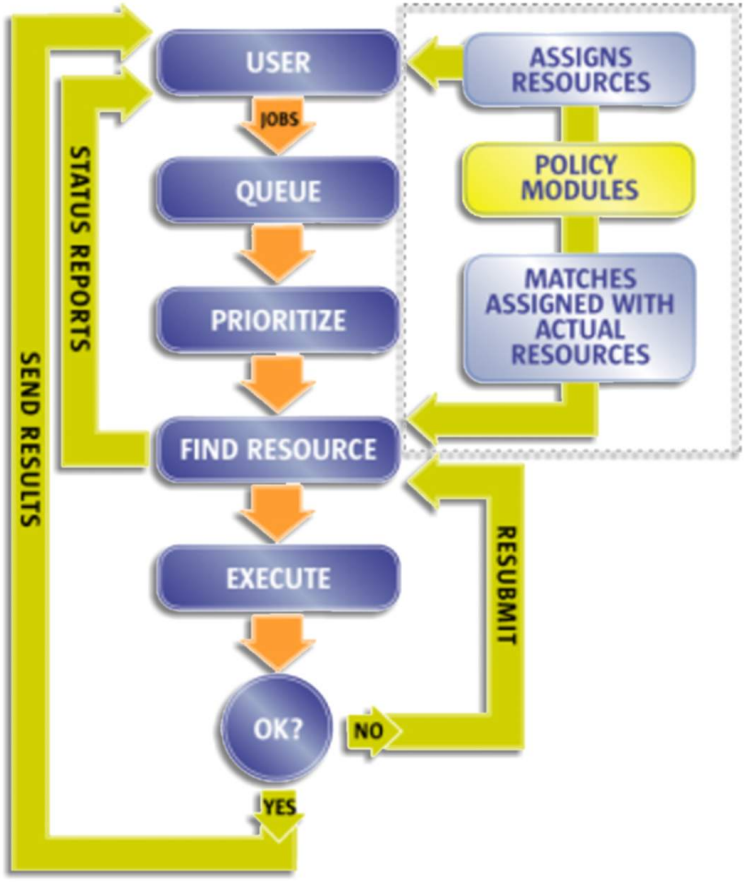
Unleashing Supercomputing to broad range of users

Science Gateways Federate Jobs across clusters

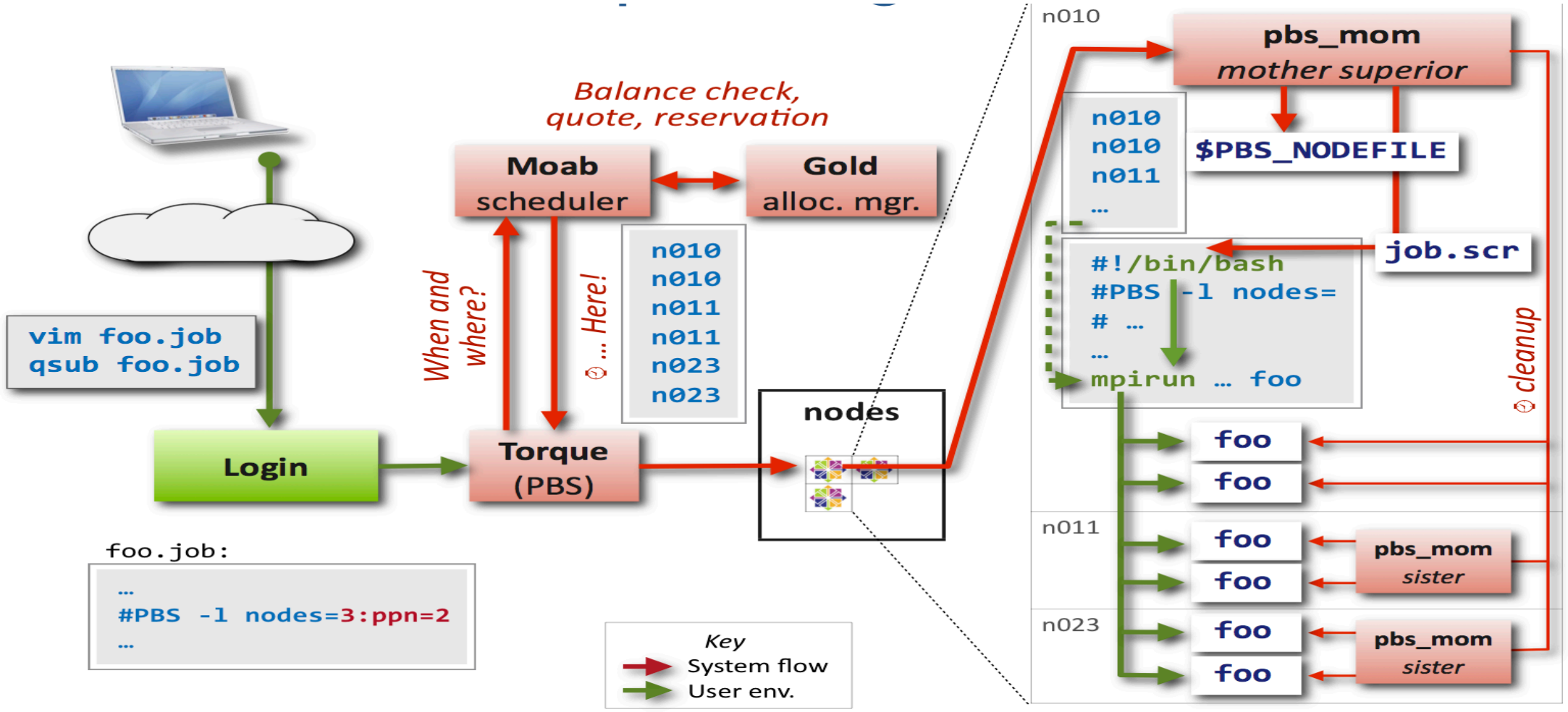


Resource Batch Managers

- Dynamic Resource Management
  - Job Scheduling
  - Resource monitoring
  - Policy administration
  - User authentication and access control
  - Accounting and reporting
- Popular Tools
  - SLURM
  - PBS



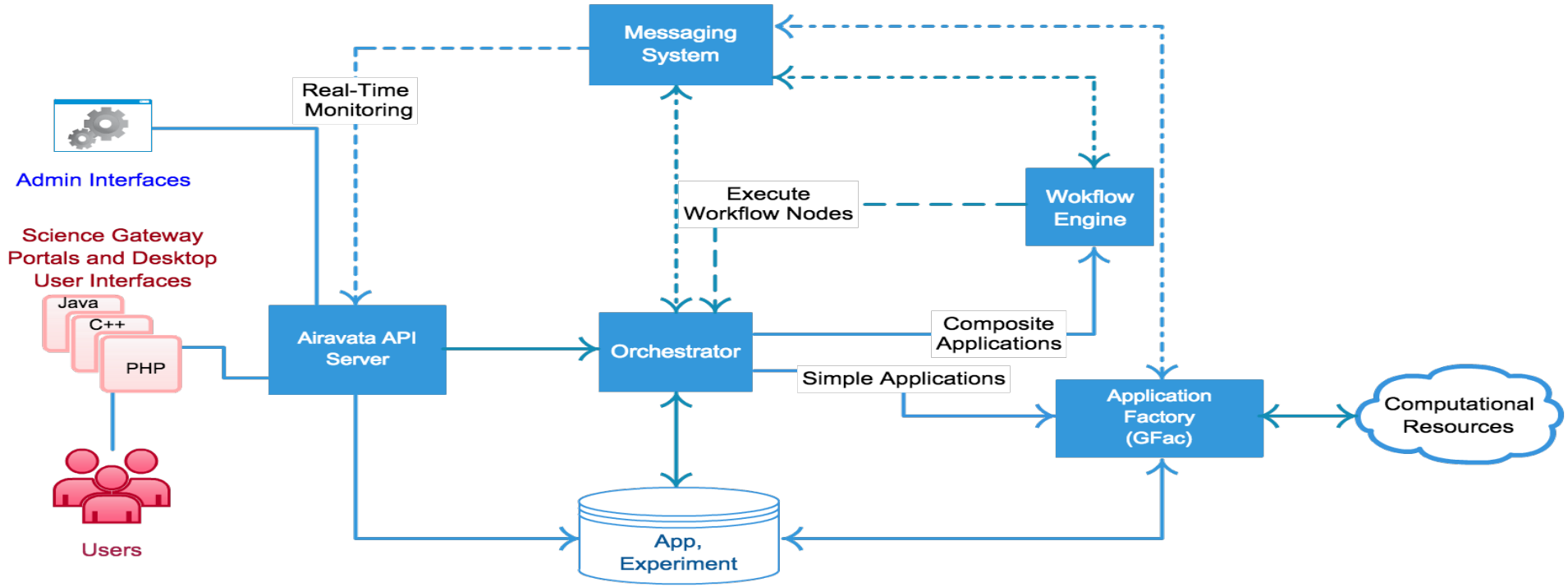
Job Managers Flow



Source: https://wiki.anl.gov/cnm/HPC/Submitting_and_Managing_Jobs

Apache Airavata

Airavata: Multi-Tentanted Gateway Middeware



- External clients interact with Airavata API (based on Apache Thrift).
- Internally, components interact with each other through Component Programming Interfaces (thrift based CPIs).

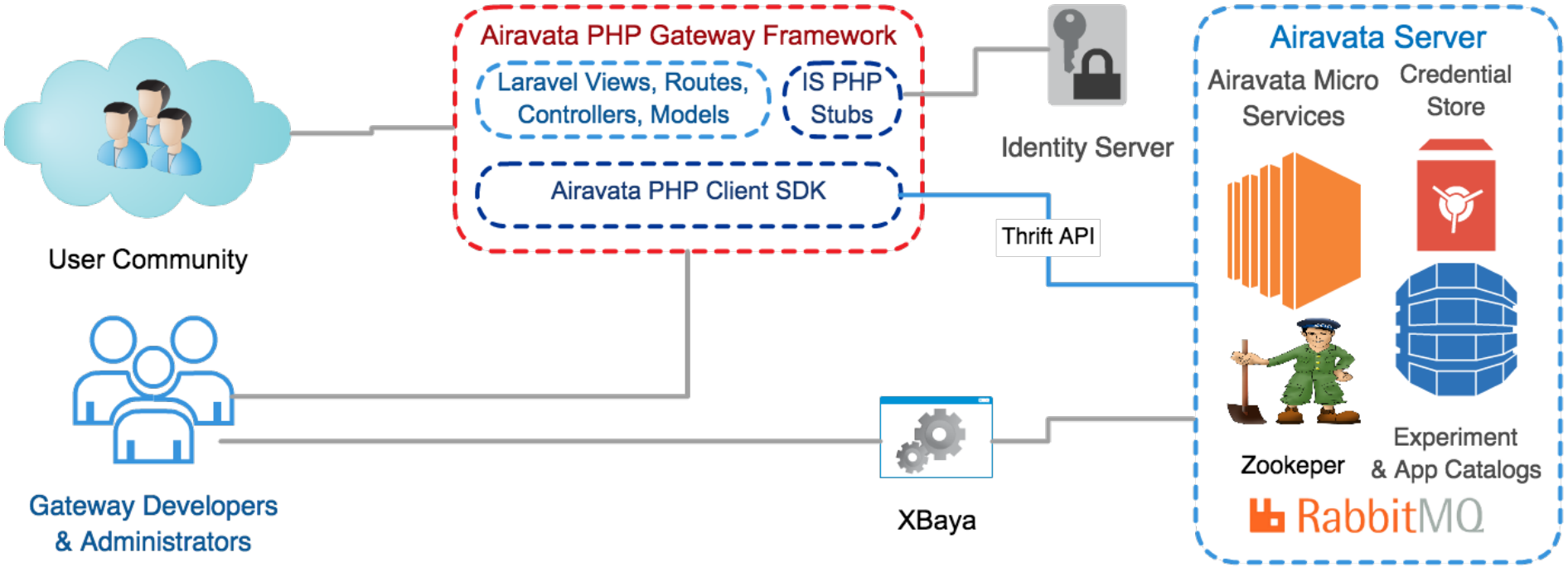
Airavata Overview

- Airavata is a general purpose distributed system software framework build on micro-service and component based architecture principles.
- Airavata provides capabilities to compose, manage, execute and monitor large scale applications and workflows on distributed computing resources.
- Airavata supports executions on local clusters, national grids, academic and commercial clouds.
- Airavata is inherently multi-tenanted.

Airavata as a Science Gateway Middleware

- Airavata is dominantly used to build science gateways.
- Airavata supports secured communications to HPC resources and empowers gateway operators to administer and monitor long running executions.
- A reference PHP based gateway is provided to illustrate the Airavata capabilities and can be used to customize science-centric gateways

Reference Gateway <-> Airavata Services

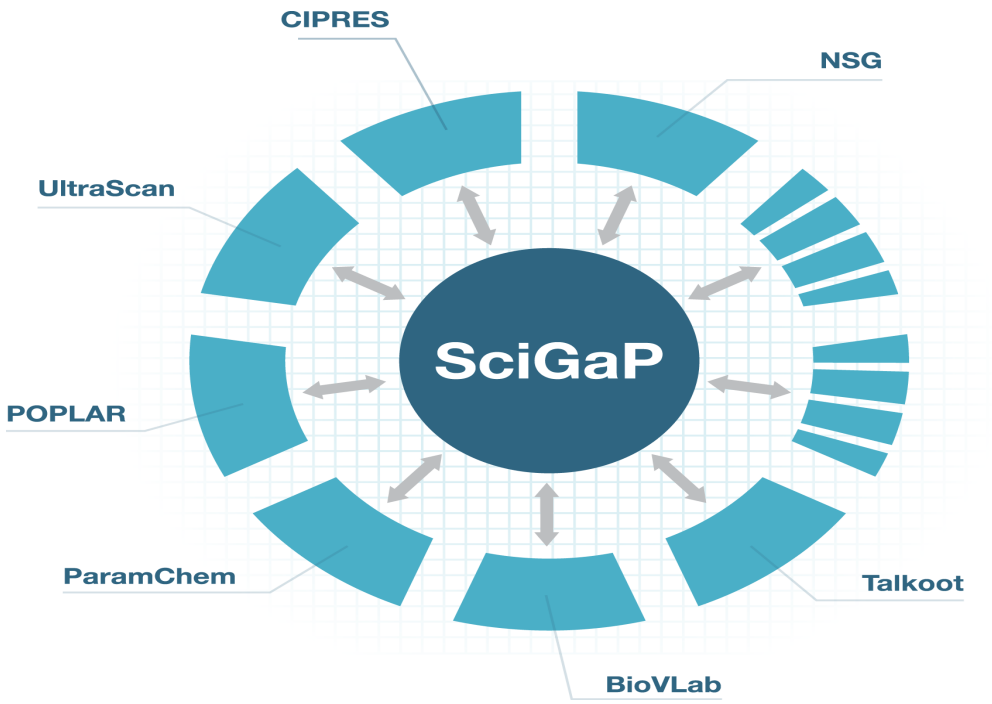


SciGaP – Powered by Airavata

- Science Gateway Platform as a Service (SciGaP) provides application programmer interfaces (APIs) to hosted generic infrastructure services that can be used by domain science communities to create Science Gateways.
- SciGaP hosted service platform is powered by Apache Airavata.
- SciGaP helps gateway developers to concentrate their efforts on building their scientific communities and not worry about operations.

SciGaP Key Mission

Scale number of gateways without having to scale FTE's needed to support them



powered by

Apache Airavata

Challenges for Gateways

- Providing a rich user experience
- Defining an API for the application server
- Defining the right sub-components for the application server.
- Implementing the components, wiring them together correctly.
- Supporting multiple gateway tenants
- Fault tolerance for components
- State management
- Continuous delivery
- Security management
- Supporting full scientific exploratory cycle

Assignments - Project Themes

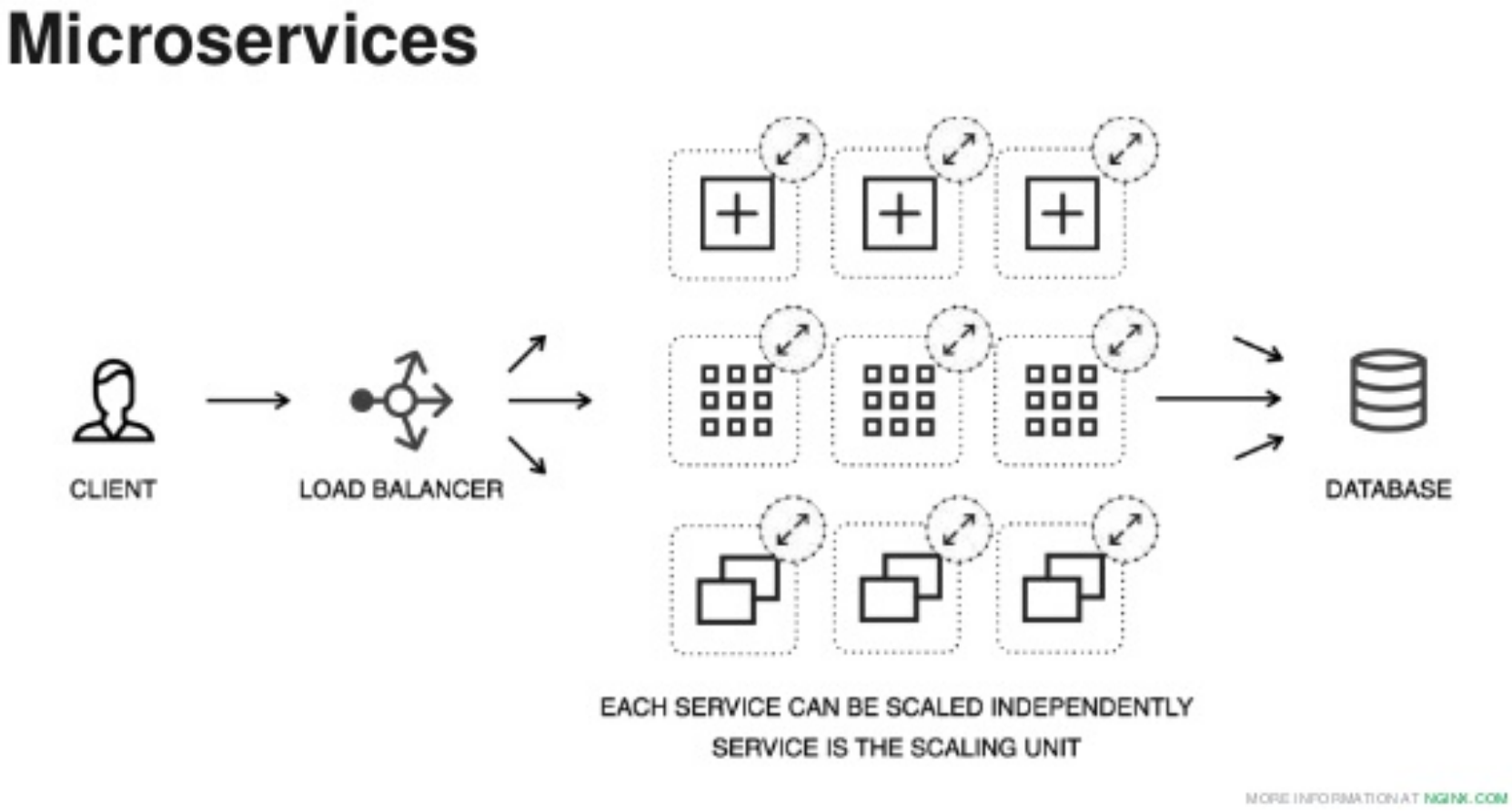
2 week increments

Project Themes

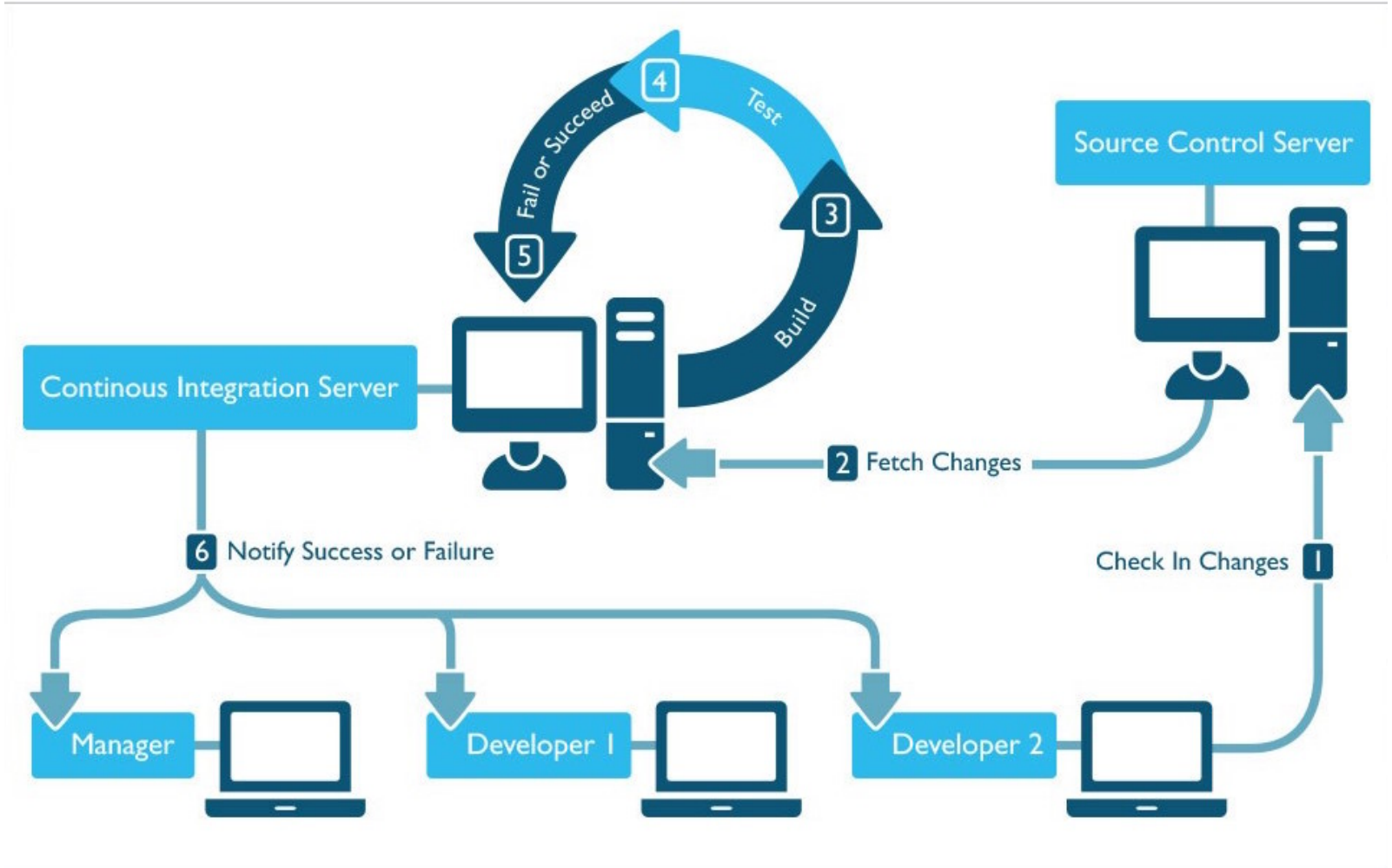
1. Theme 1: DevOps
2. Theme 2: Code Contributions to Airavata

Theme 1: DevOps

Component Microservices - Scaling

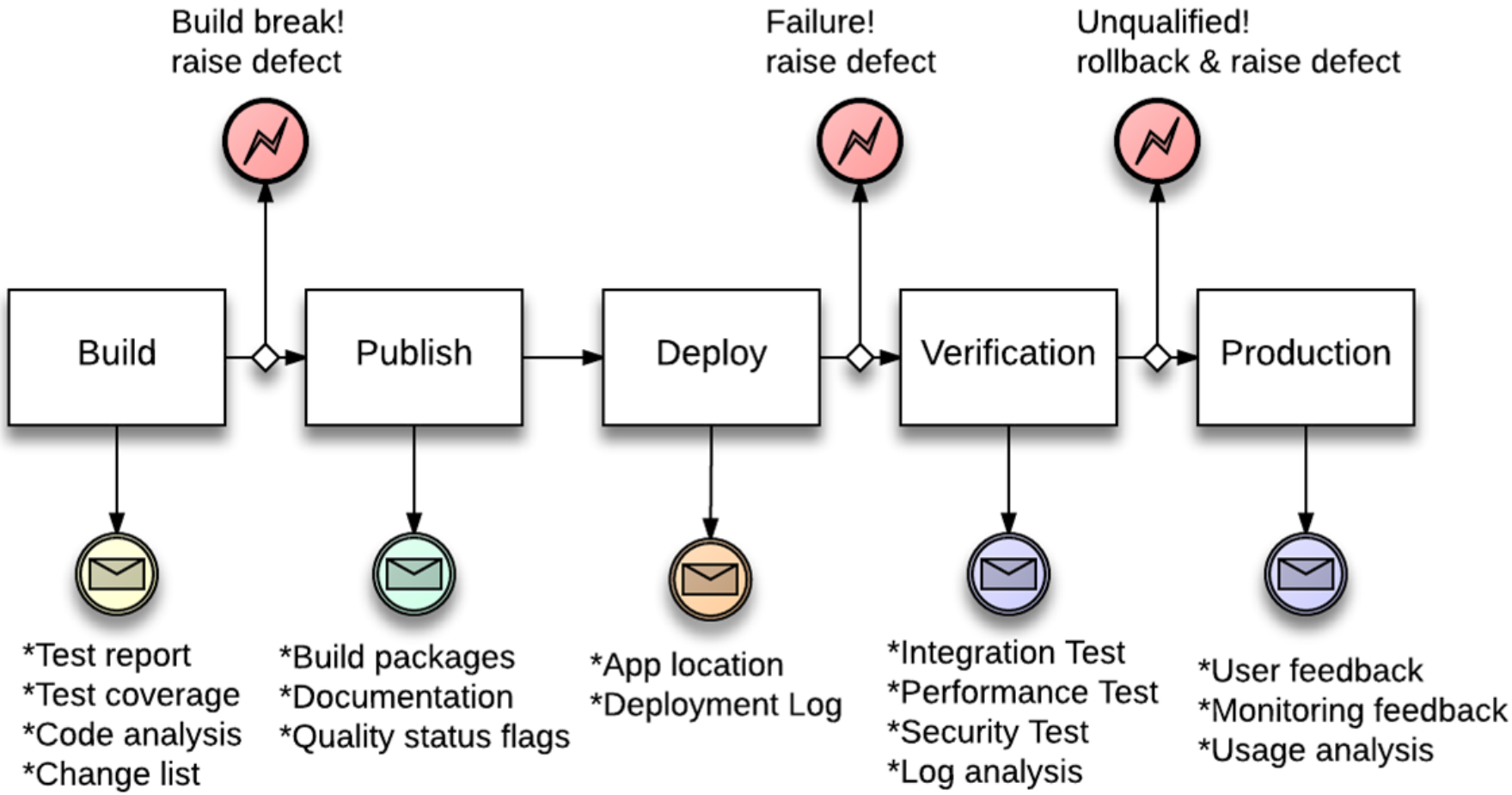


Continuous Integration



Source: https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html

Continuous Deployments/Delivery



Source: https://www.ibm.com/developerworks/community/blogs/c914709e-8097-4537-92ef-8982fc416138/entry/devops_in_practice_best_practices_for_adopting_continuous_delivery?lang=en