Galapagos:
a generic distributed parallel genetic algorithm development platform

Nicolas Kruchten
4^{th} year Engineering Science
(Infrastructure Option)

Last Week

- Introduction to Galapagos:
  - Background
  - Motivation
  - Capabilities
  - What but not how…

Major Points

- Galapagos is a platform with which we can develop distributed parallel genetic algorithms
- Galapagos is very flexible
  - Many (GA-appropriate) problems
  - Many GA types
    - Operators
    - Population structure
  - Many distribution schemes

Today

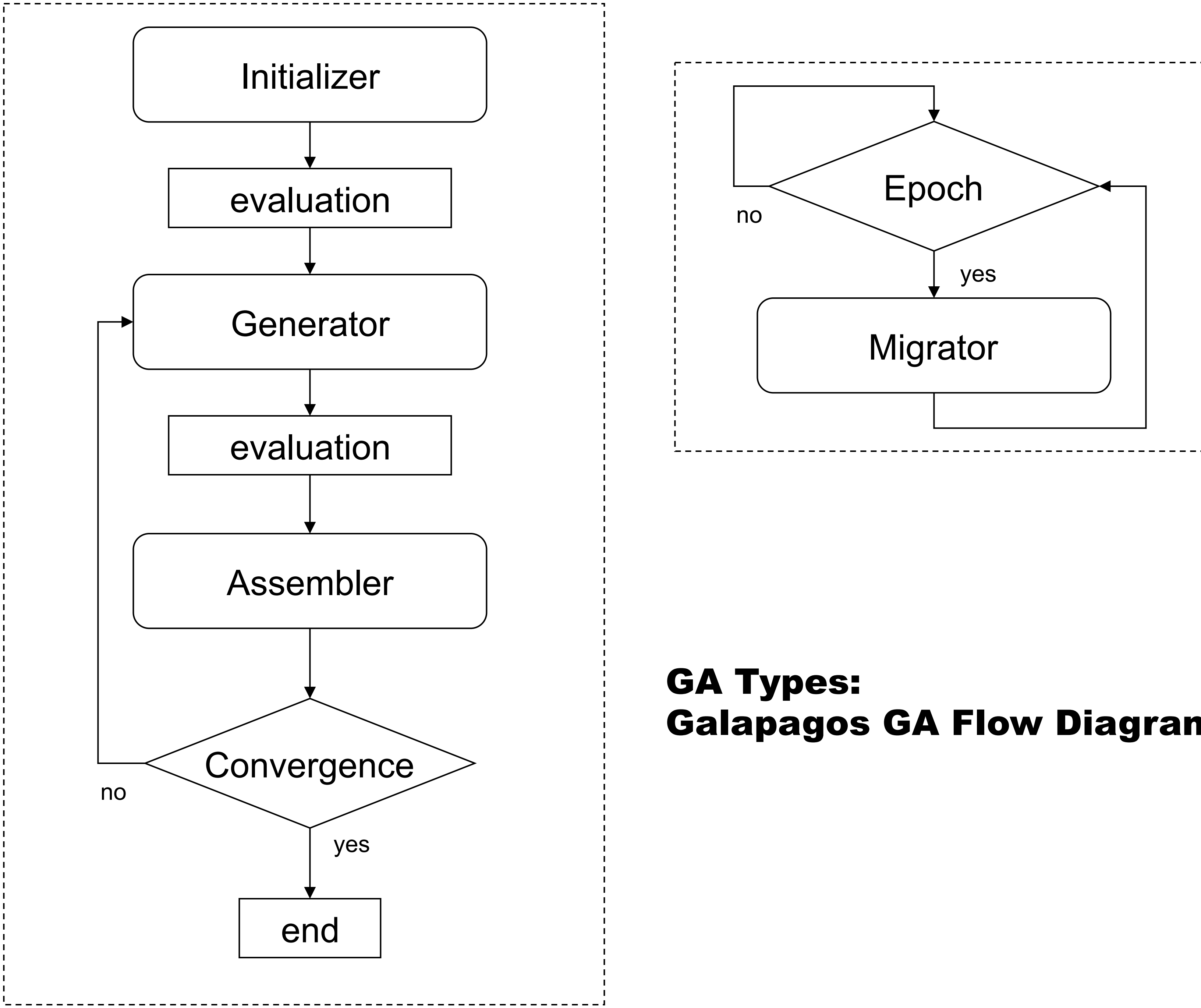
- Technical explanation of Galapagos
- Approach: 3 views of Galapagos' flexibility
  - Problem-domain
  - GA types & parallel population structure
  - GA distribution architecture

Problem-Domain

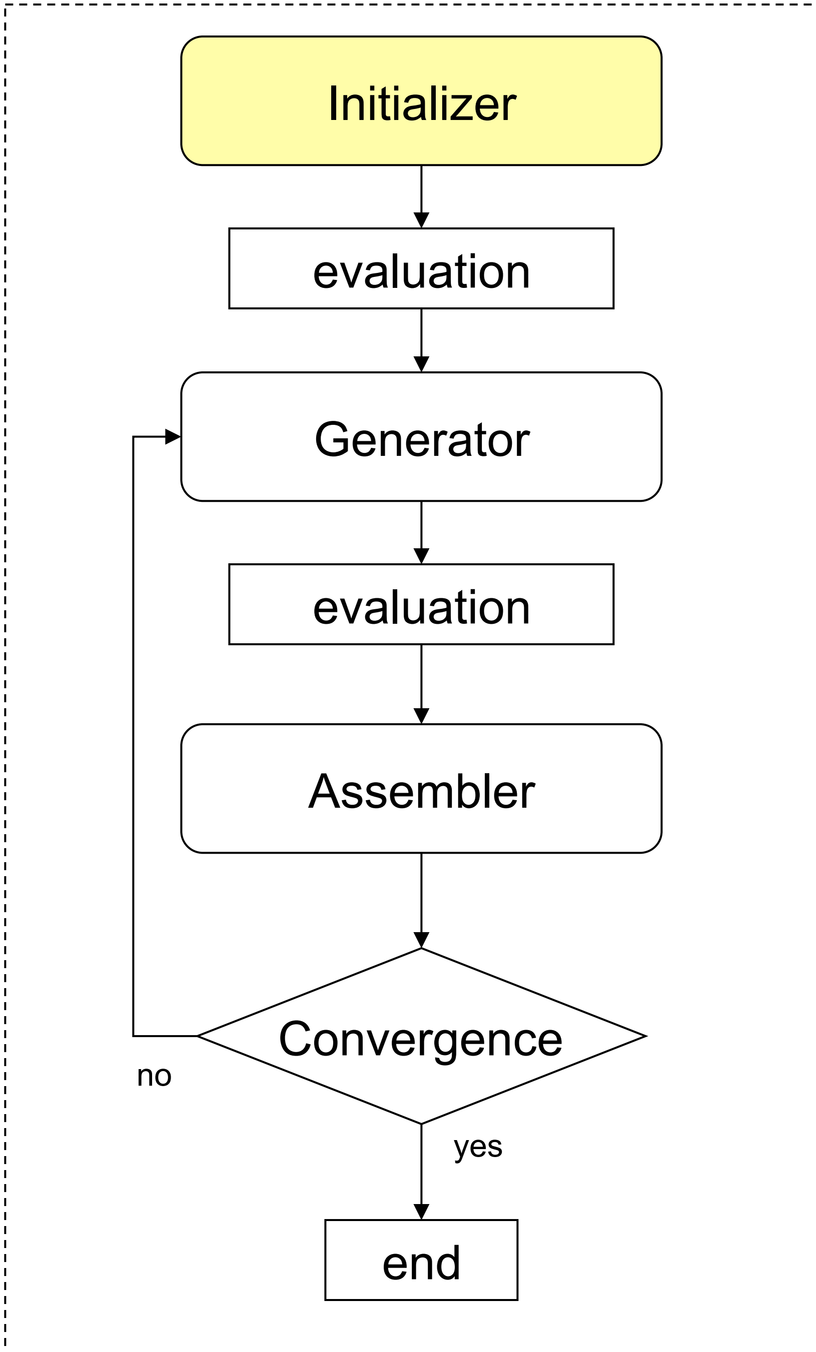
- GAs are optimization tools
- Need objective function
  - Not always real function of real variables
- Galapagos can be used with a wide variety of objective functions

Object Oriented

- Galapagos defines and executes a set of ‘black-box’ steps in a flowchart
- Internal workings of boxes, and data types they operate on are up to the developer
- Galapagos provides templates for building new data types and GA operators

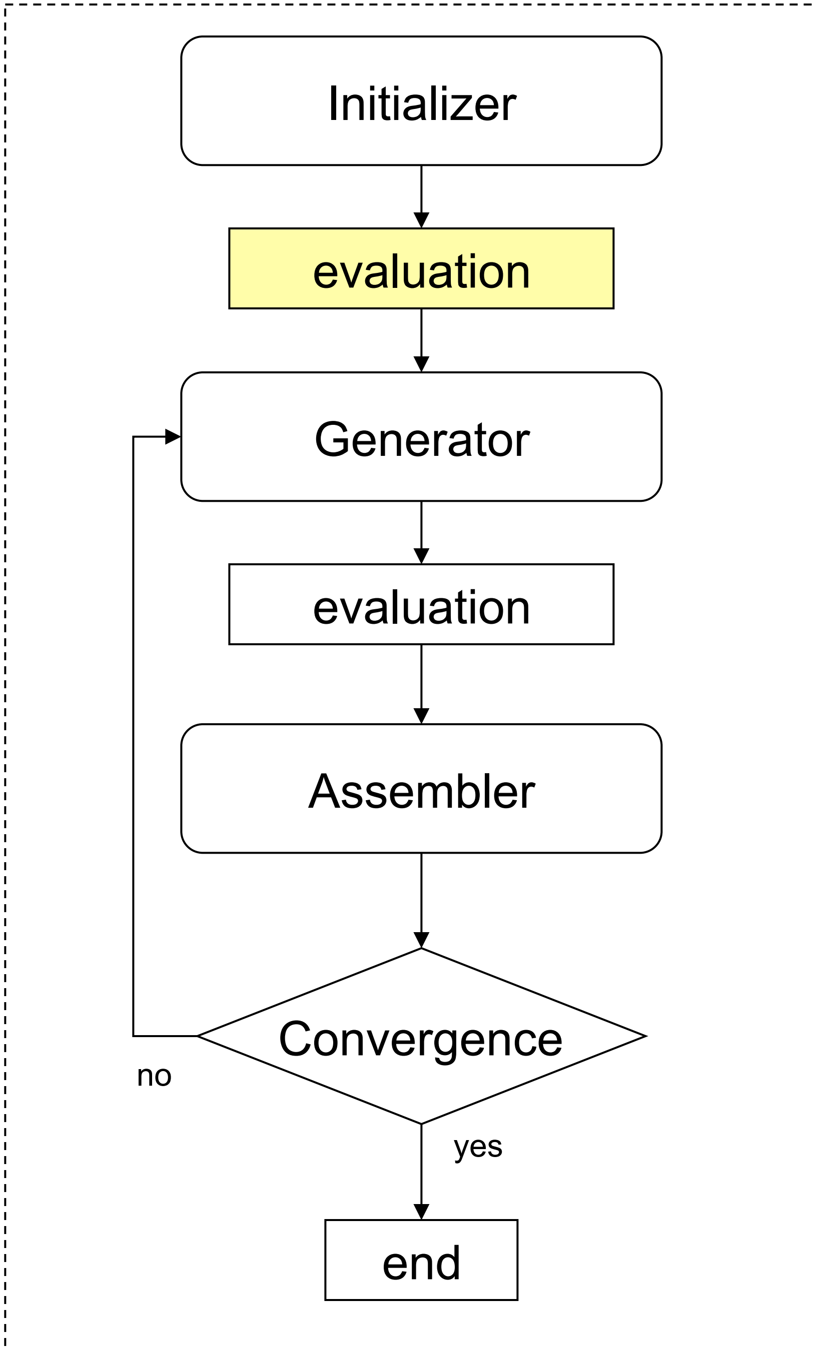


GA Types:
Galapagos GA Flow Diagram



Initializer

- picks the initial population
- default: random, specified min/max
- other uses: seeding the population

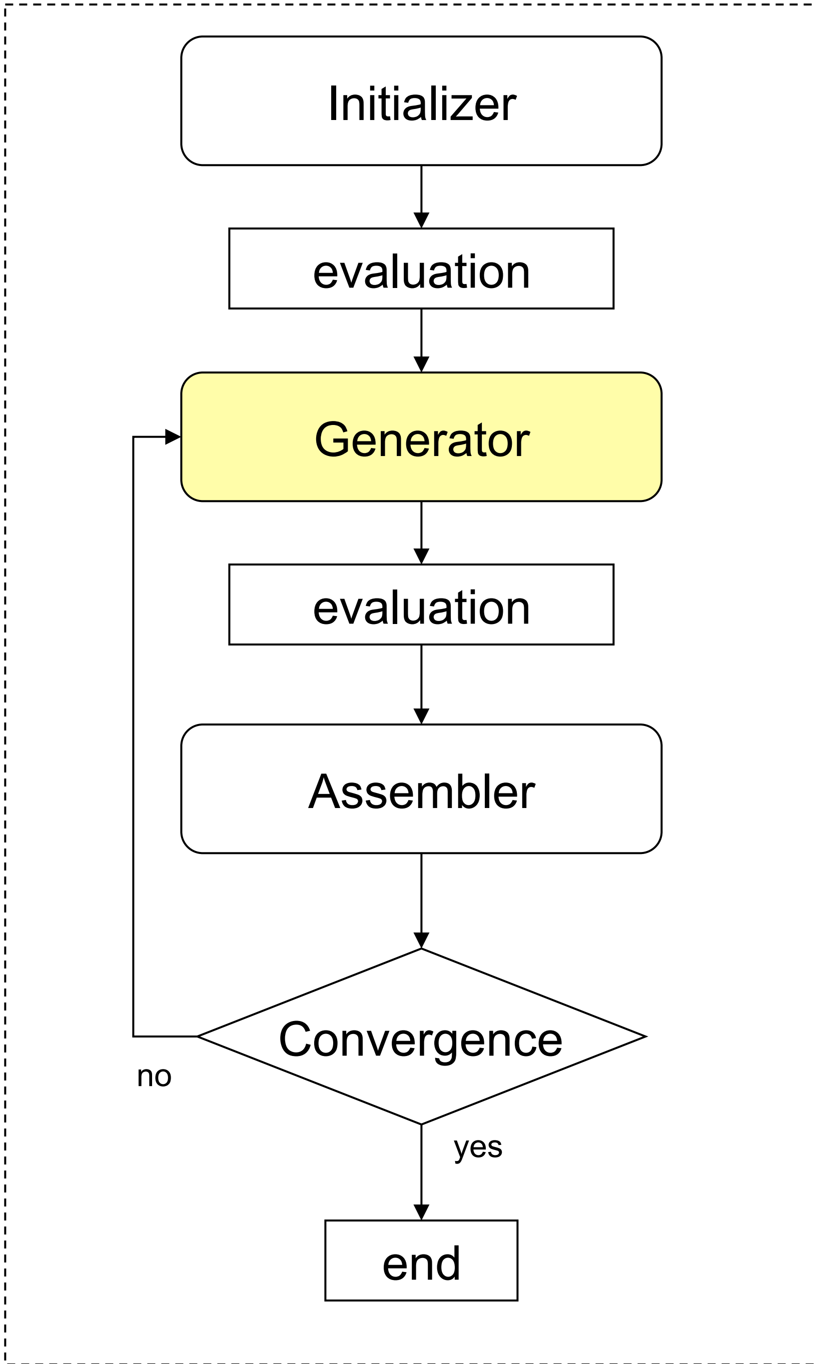


evaluation

• handled by Galapagos

• usually handed to LightGrid for distributed processing

• can be handled on local machine (single-computer version)

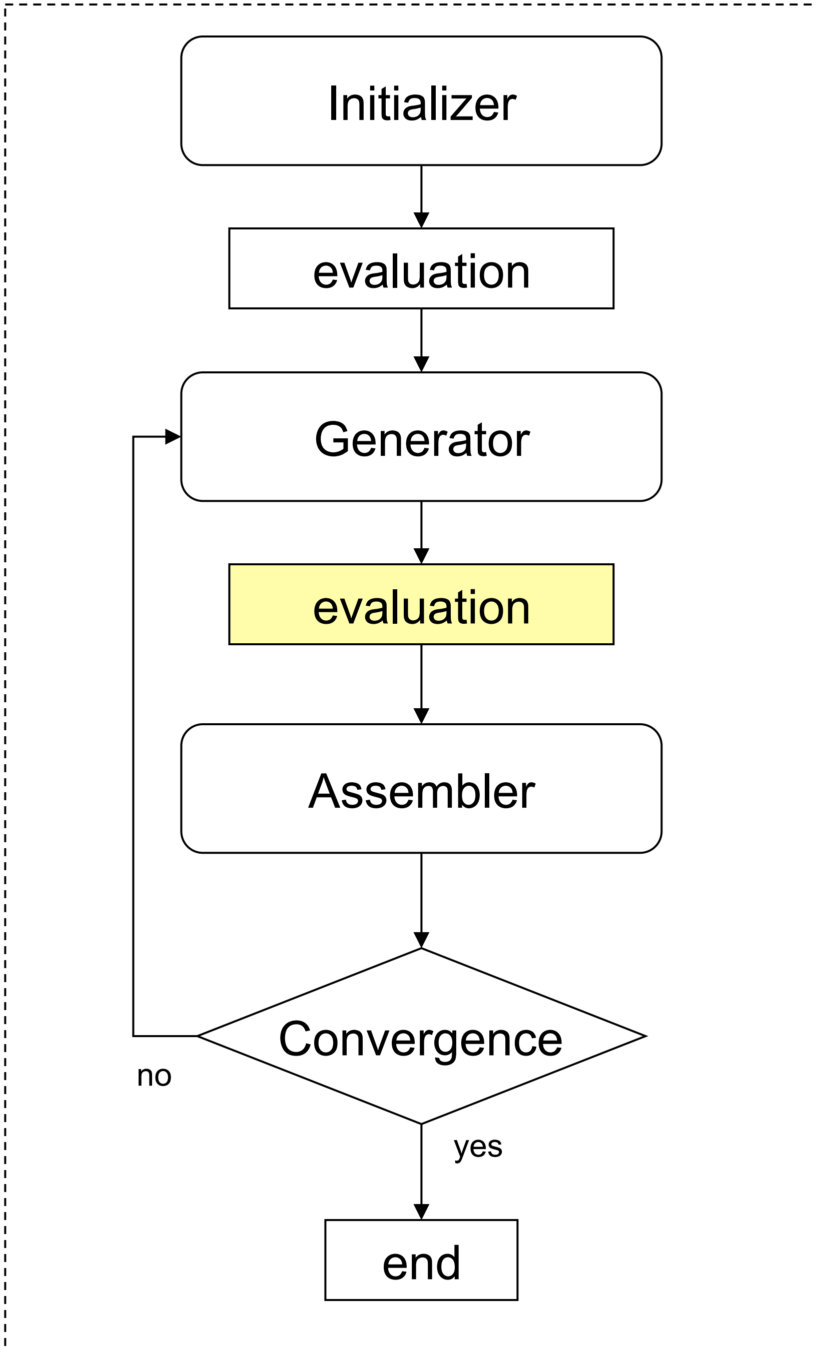


Generator

- Creates a generation of children
- GA *usually* works with:
    - A Mutator
    - A Recombiner (compare: EP)

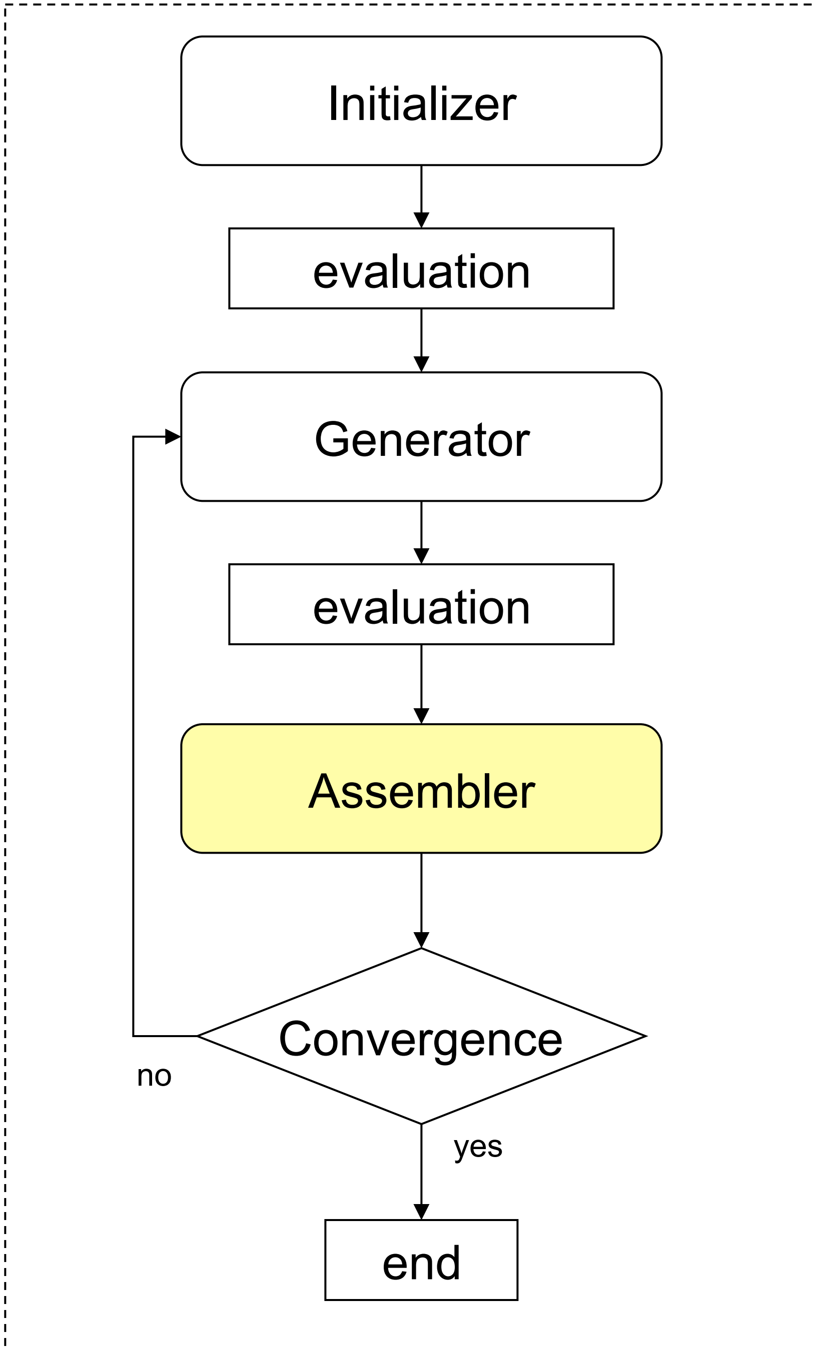
each child = mutate(recombine())

(mutate the output of recombination operation)



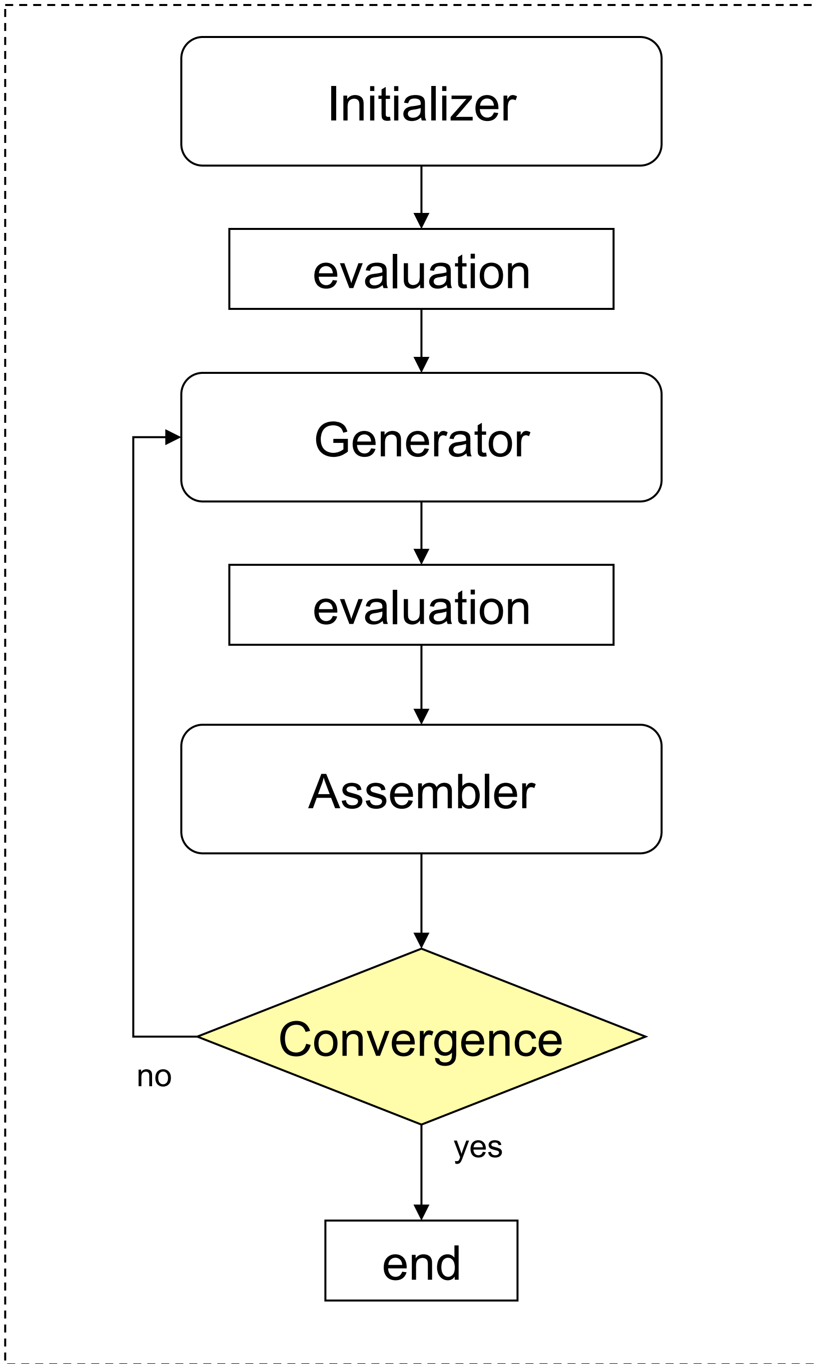
evaluation

- same as above



Assembler

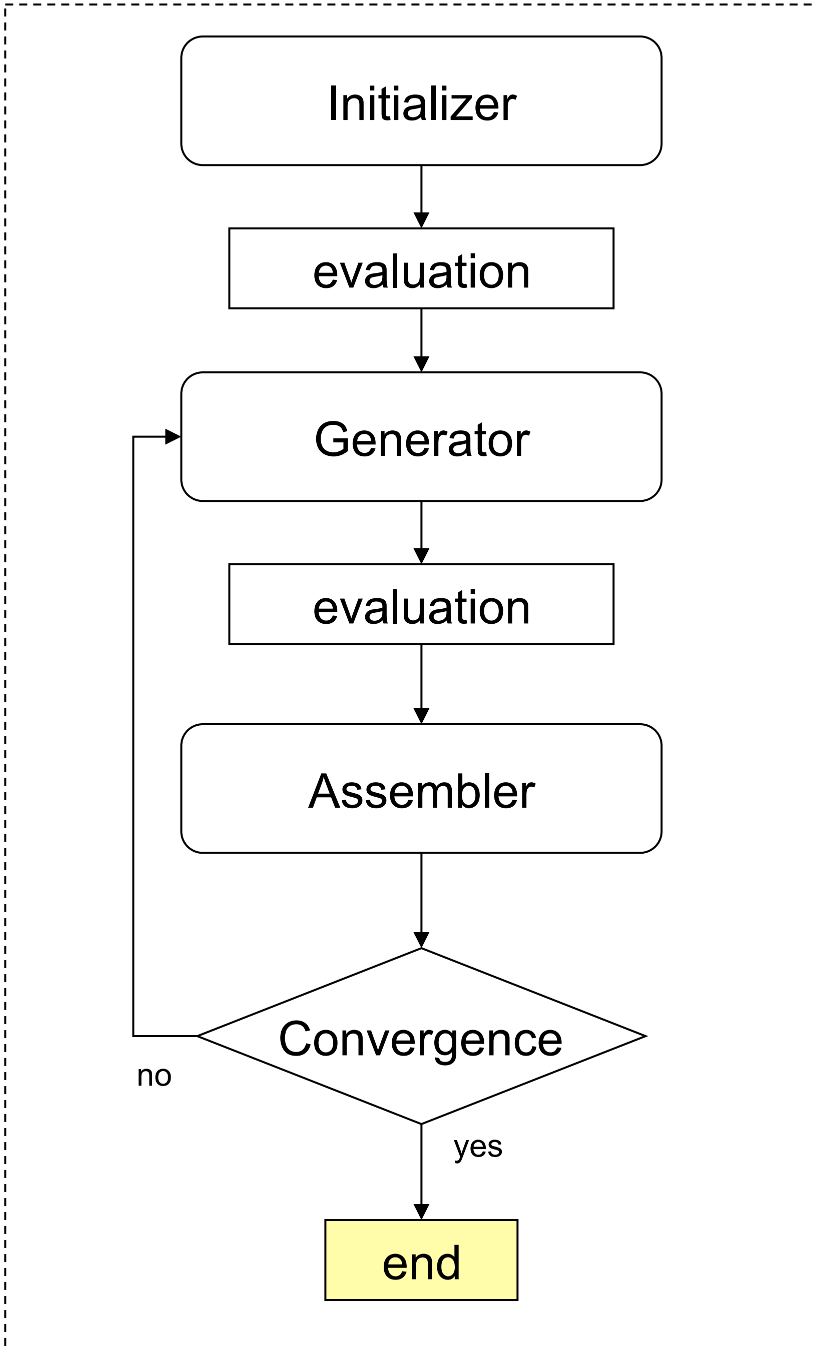
- Creates a new population, using an existing population and an incoming generation
- GA falls roughly in:
    - Simple ('canonical')
    - Crowding ('steady-state', 'incremental', 'truncation')
- Crowding *usually* implemented using a Selector
- This is where elitism occurs



Convergence

- Defines some end condition for the run

- Usually based on:
    - A real time measure
    - An 'algorithm time' measure
    - Some parameter (mean, std etc.)



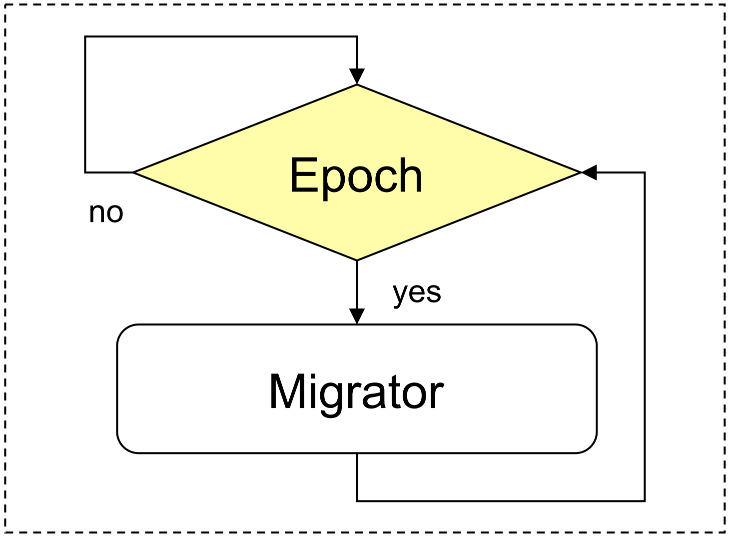
end

- Run is over
- Whatever post-processing, logging etc happens here

Epoch

- Asynchronously evaluated

- Defines some timeframe:
  - Real time
  - 'algorithm time'
  - Some parameter (mean, std, etc.)



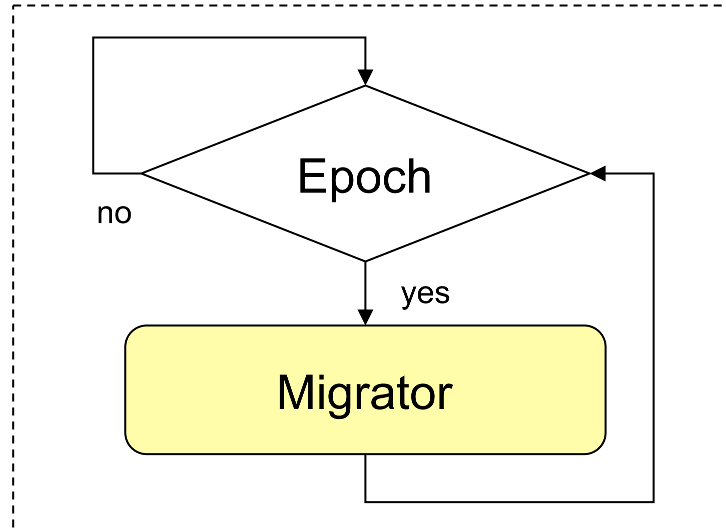
Migrator

- Required for Parallel GA (multiple populations)

- On sending side, defines:
    - migrant destination
    - migrant number
    - migrant selection

- On destination side, defines:
    - replacement policy

- *usually* implemented with:
    - Topology
    - Selector
    - Assembler



Supporting Operators

- Generator:
  - Mutator
  - Recombiner
- Assembler:
  - Selector
- Migrator:
  - Topology
  - Selector
  - Assembler (already covered)

Mutator

- Basic EA staple
- Input: a chromosome
- Output: a chromosome
- Function: performs some mutation
- *Usually* implemented using a GeneMutator, in which case Mutator selects *which* genes are mutated, GeneMutator governs how that occurs

GeneMutator

- Usually called from a Mutator
- Input: a gene
- Output: a gene
- Function: mutate that gene according to some rule

Recombiner

- Input: none
- Output: a chromosome
- Function: uses an internal Selector to pick parents, creates n children according to some rule
- If no recombination/crossover occurs, this is EP, not GA

Selector

- Shows up everywhere!
- EA staple
- Input: selection pool, n
- Output: n chromosomes
- Function: selects n chromosomes according to some rule

Topology

- Required in Parallel GA
- Defines migrant destination, number per destination

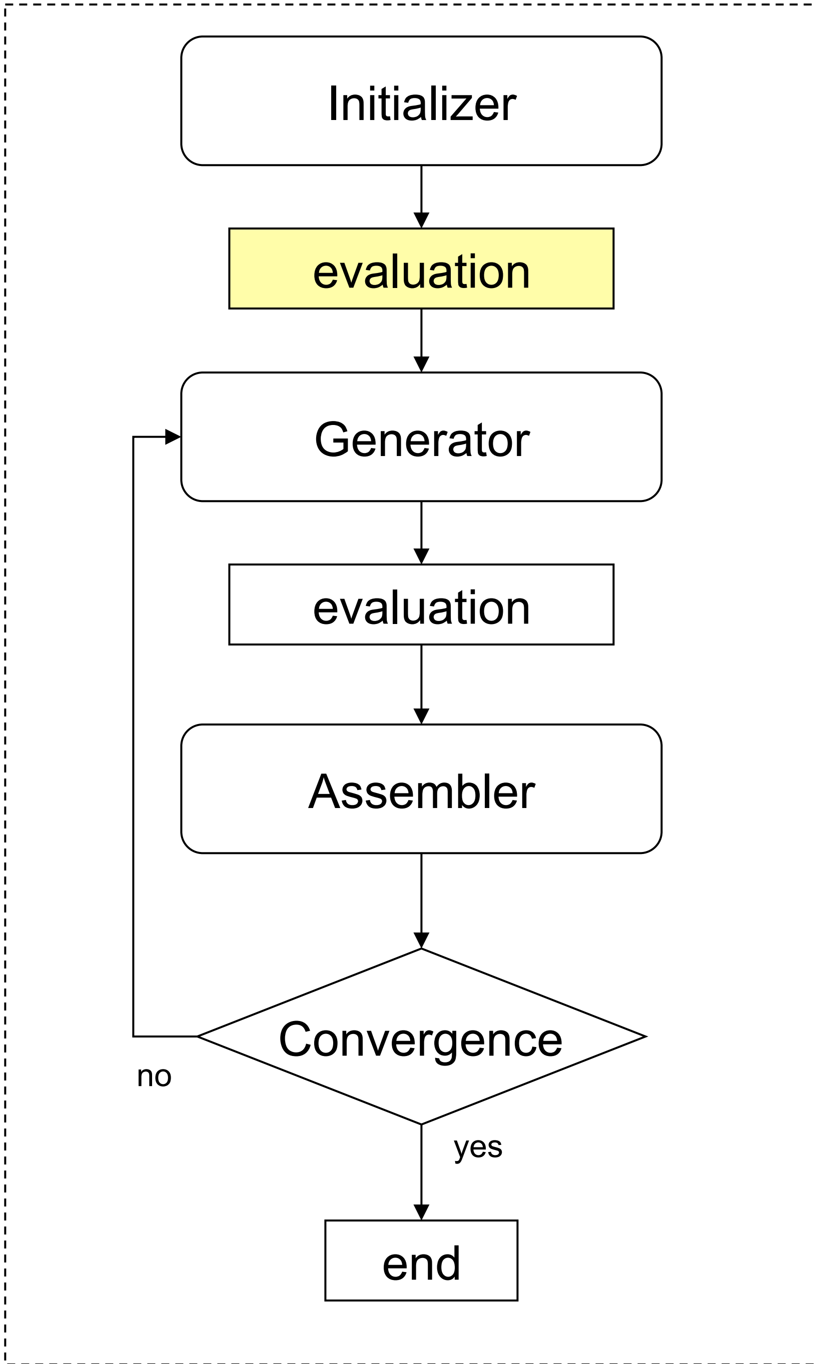
- Defines nature of PGA:
  - Coarse-grained/Fine-grained
  - Deterministic /Stochastic

Operators

- Initializer
- Generator
- Assembler
- Convergence
- Epoch
- Migrator
- Mutator
- GeneMutator
- Recombiner
- Selector
- Topology

Summary

- Meta-operators and supporting operators provide a well-defined yet flexible GA base
- Migrator/Epoch/Topology operators allow for wide variety of PGA types



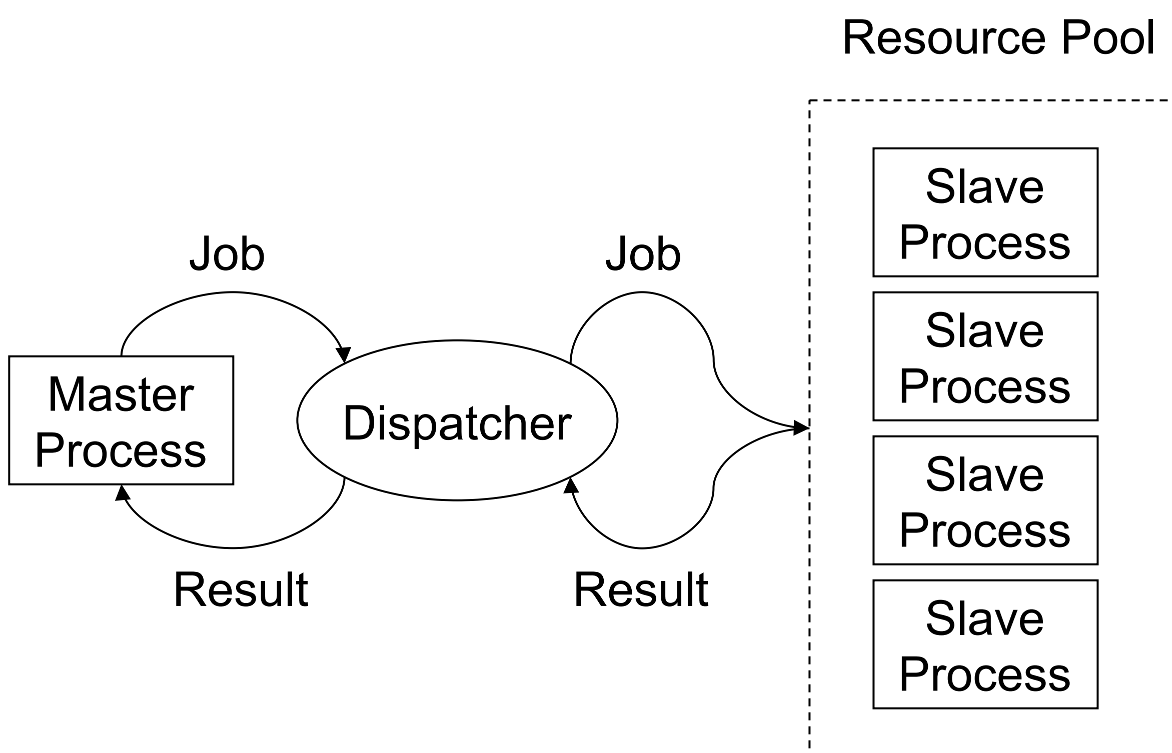
evaluation

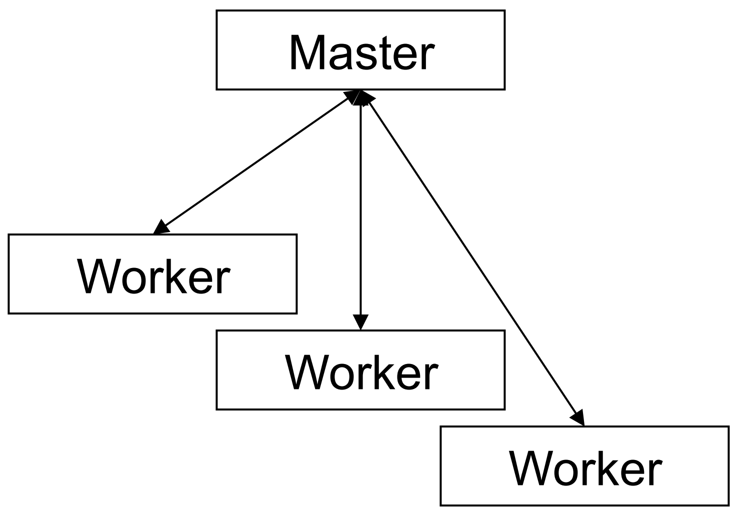
- handled by Galapagos
- usually handed to LightGrid for distributed processing
- can be handled on local machine (single-computer version)

LightGrid explained

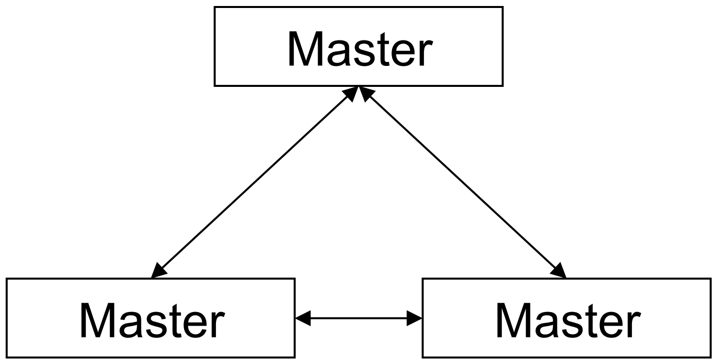
- Clients and Resources
- Dispatcher
- Generic: can be used to implement any distributed application

Grid Computing

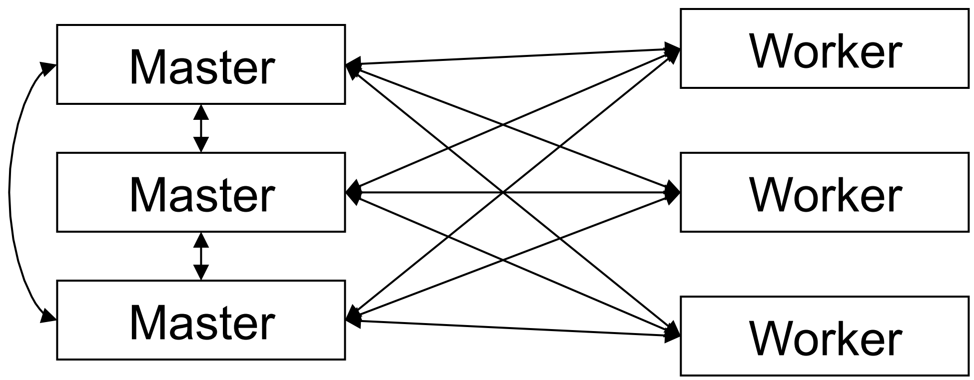




Master / Worker

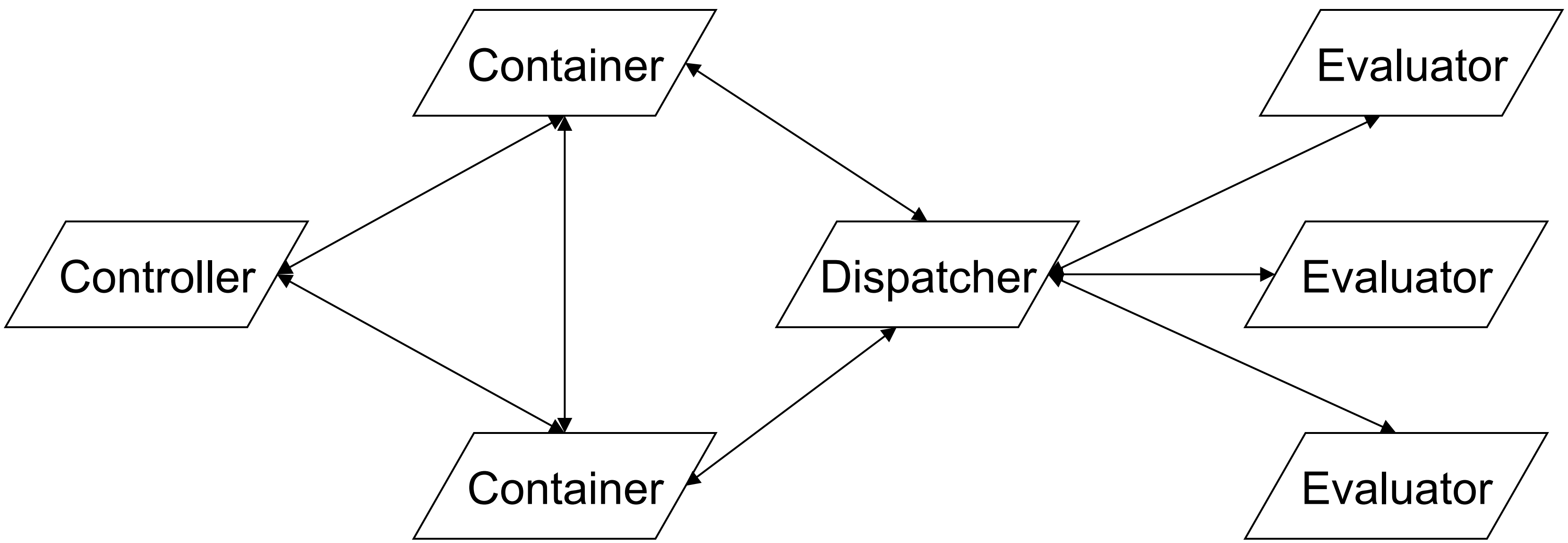


Peered



Hybrid: Peered Masters with Worker Pool

High-Level Process Diagram



Process view

- 4 major process types:
  - Container (LightGrid Clients)
  - Evaluator (LightGrid Resources)
  - Controller
  - LightGrid Dispatcher

- Each process holds more than one thread of execution (sub-process)

Process View II

- Containers (LightGrid Clients):
  - Contain and manage populations
  - Interface between populations and evaluators
  - Interface between individual populations

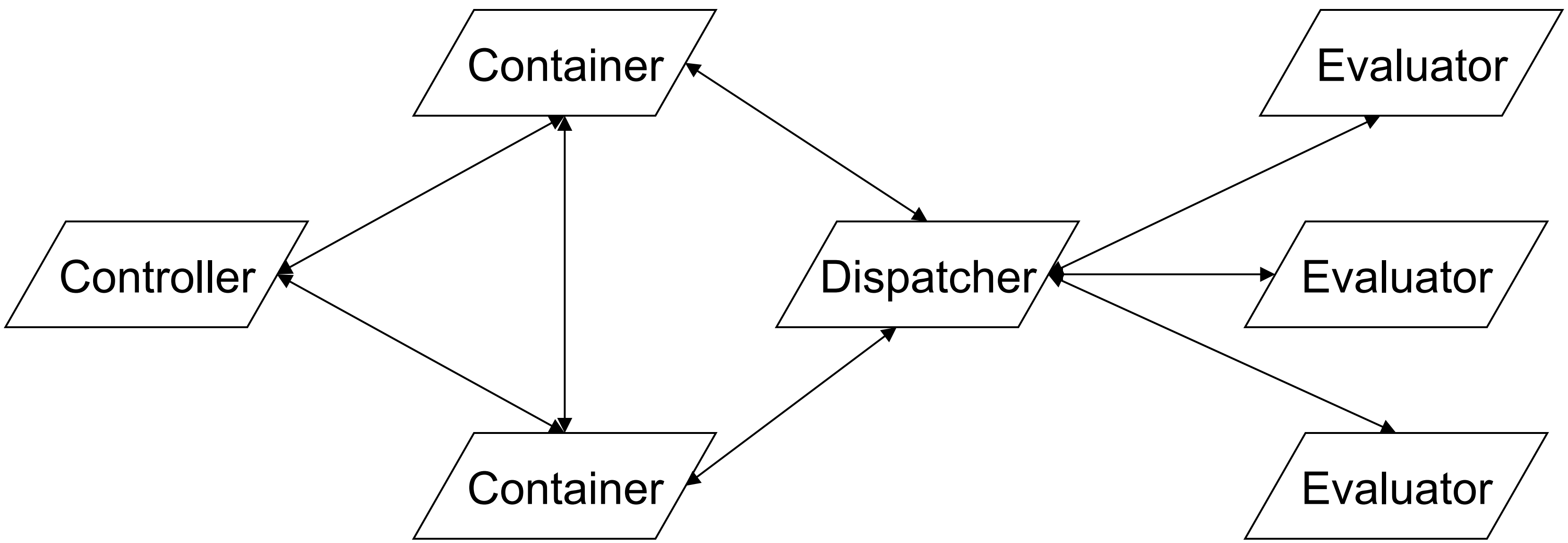
- Evaluators (LightGrid Resources):
  - Compute the fitness value for a given chromosome

Process View III

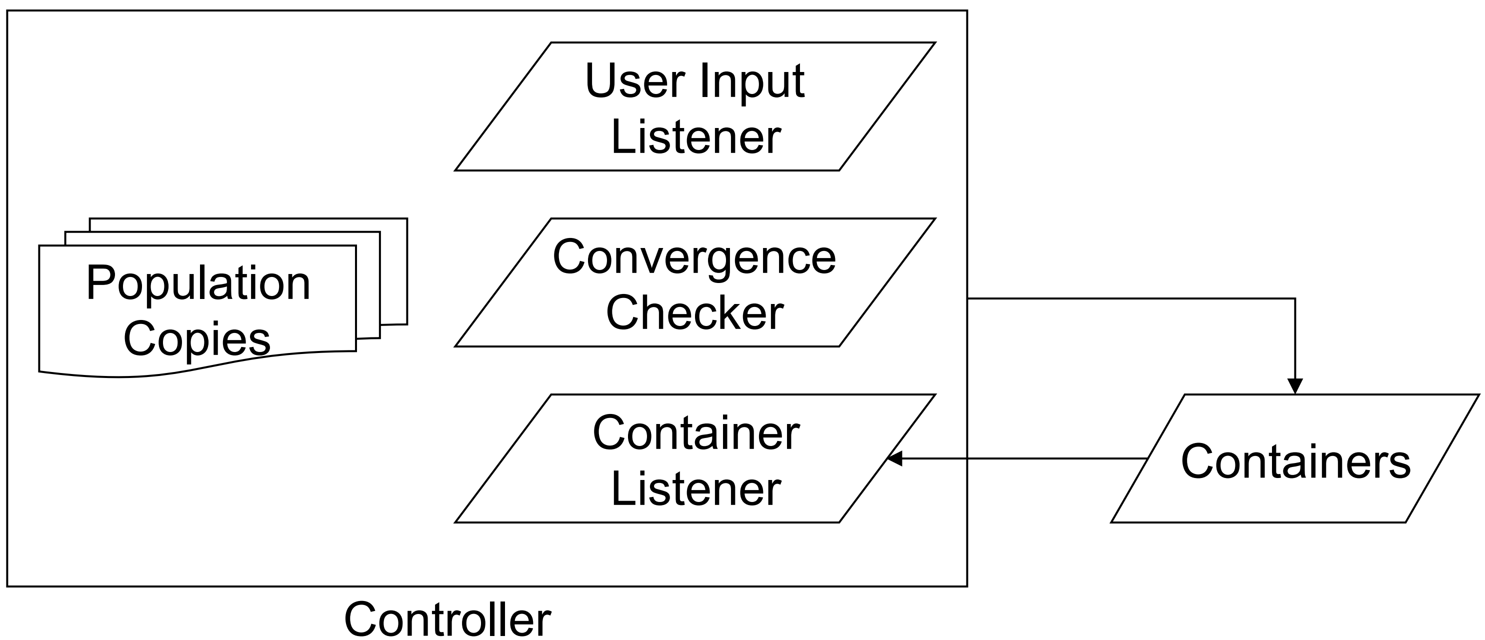
- Controller:
  - 'where the user sits'
  - Input/Output, control process
  - Issues start/stop/reset messages

- LightGrid Dispatcher:
  - Dispatches jobs from Clients to Resources and passes the results back

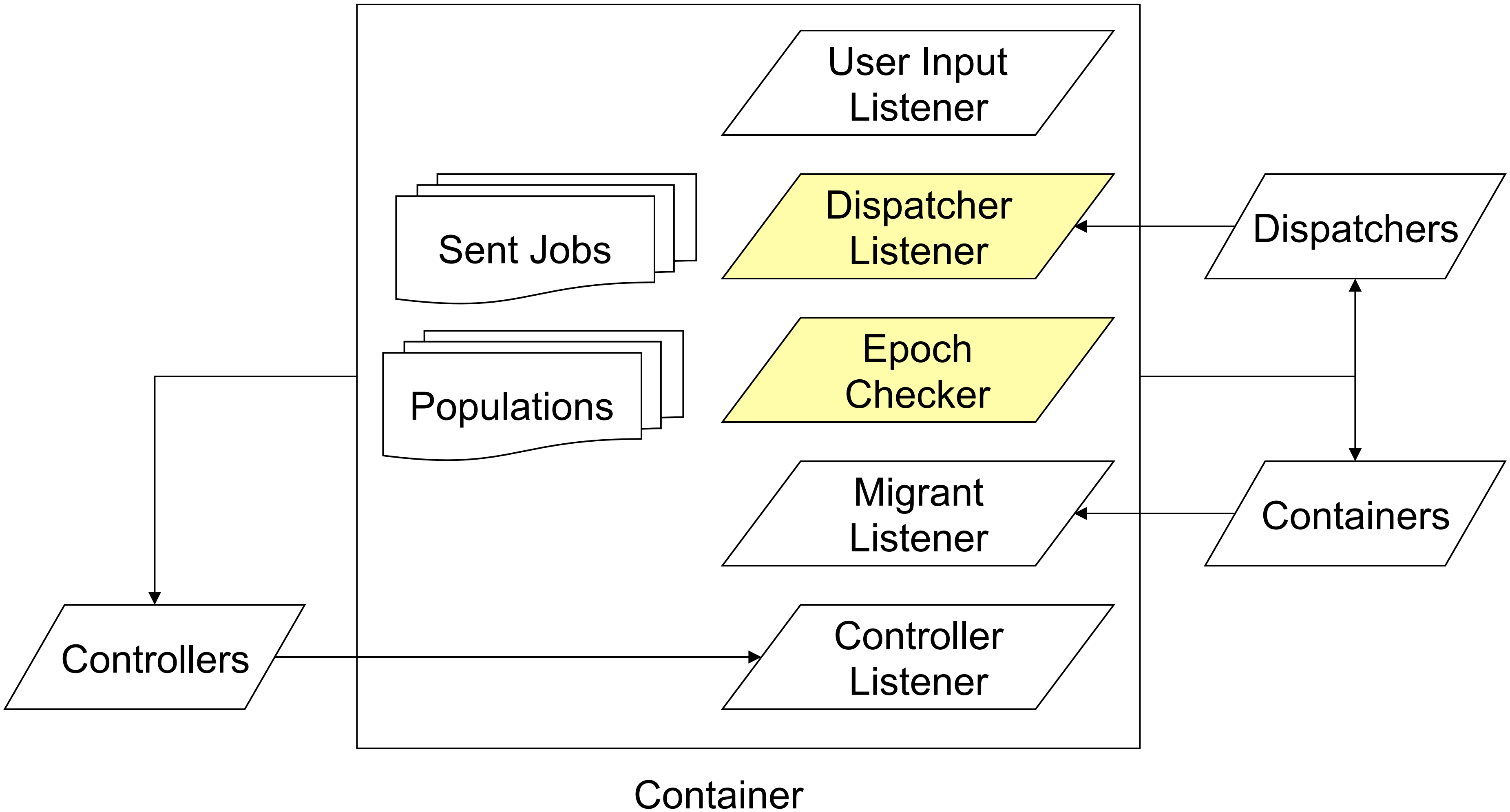
High-Level Process Diagram



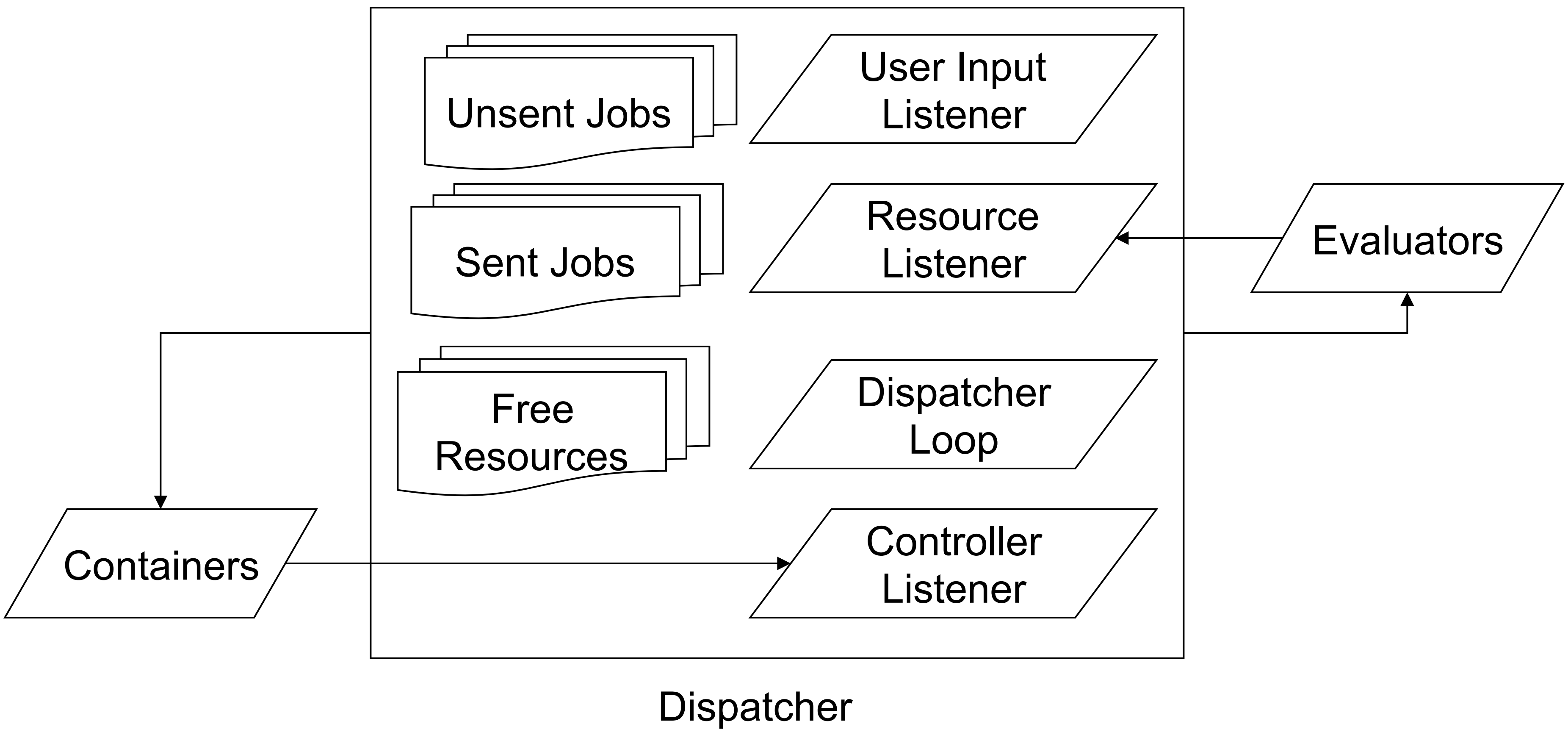
Controller Process Diagram



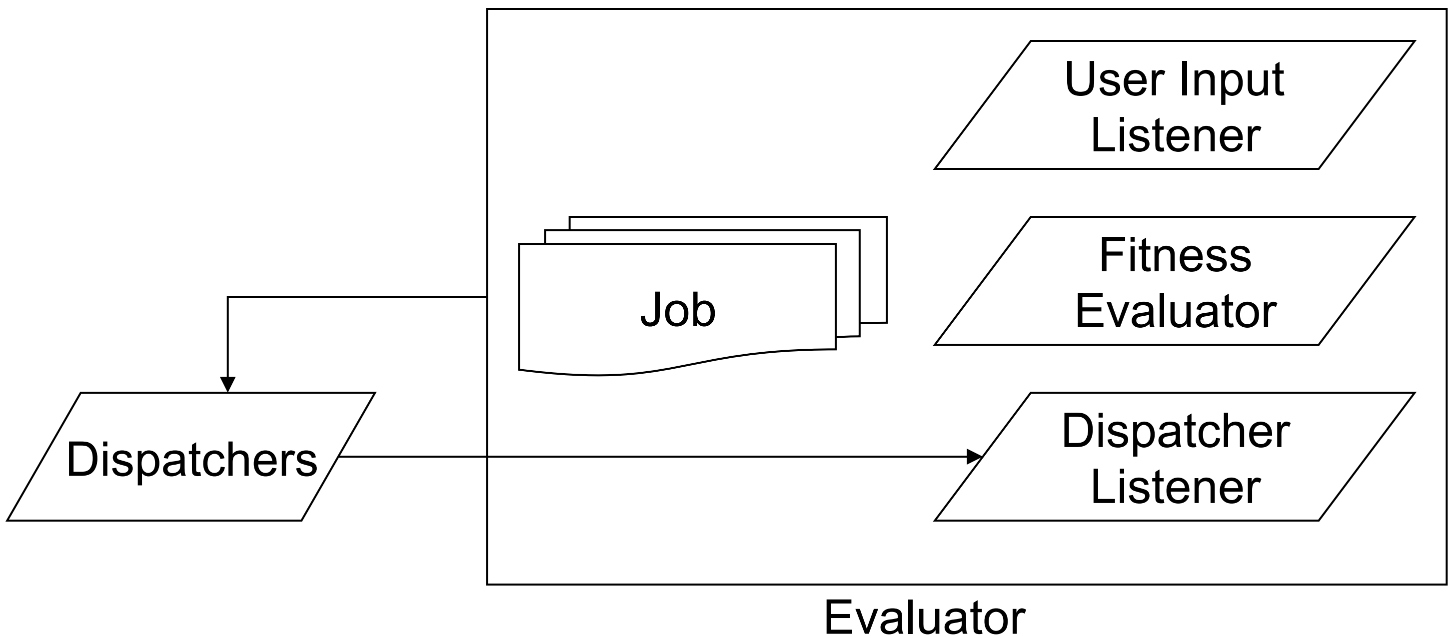
Container Process Diagram



Dispatcher Process Diagram



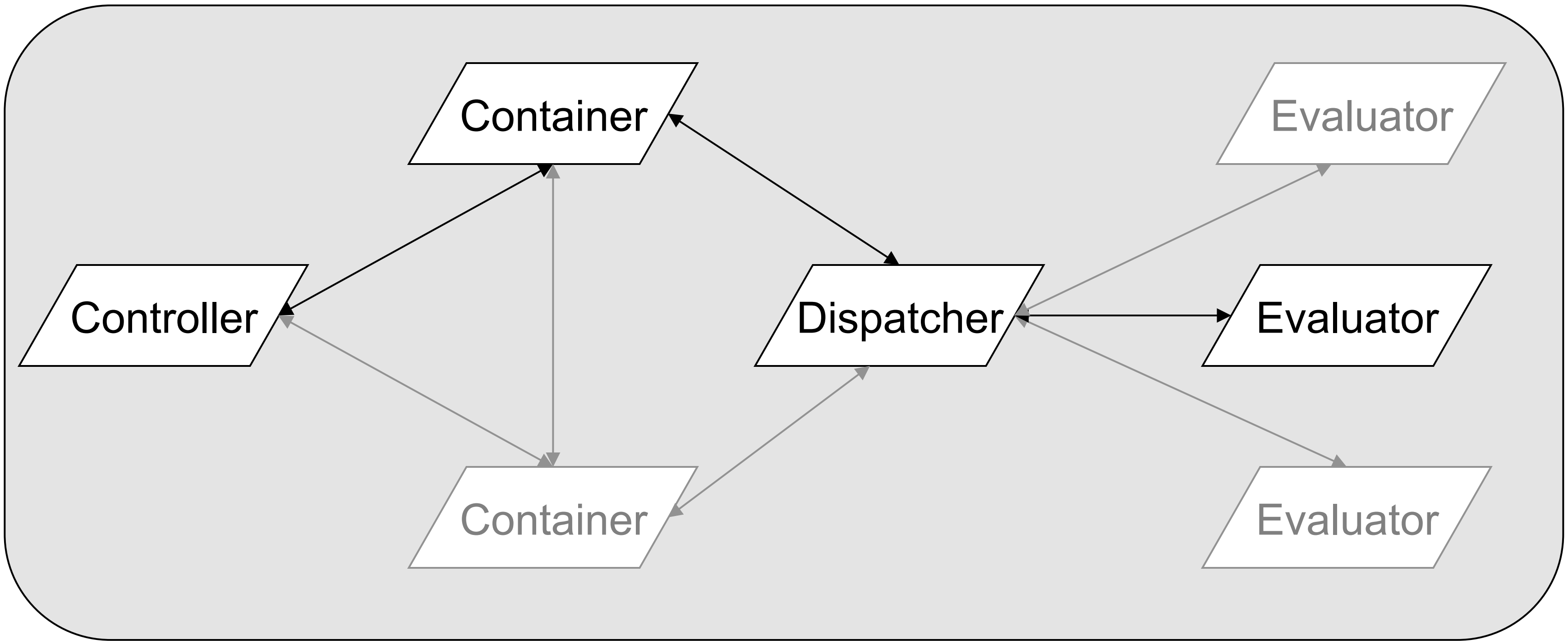
Evaluator Process Diagram



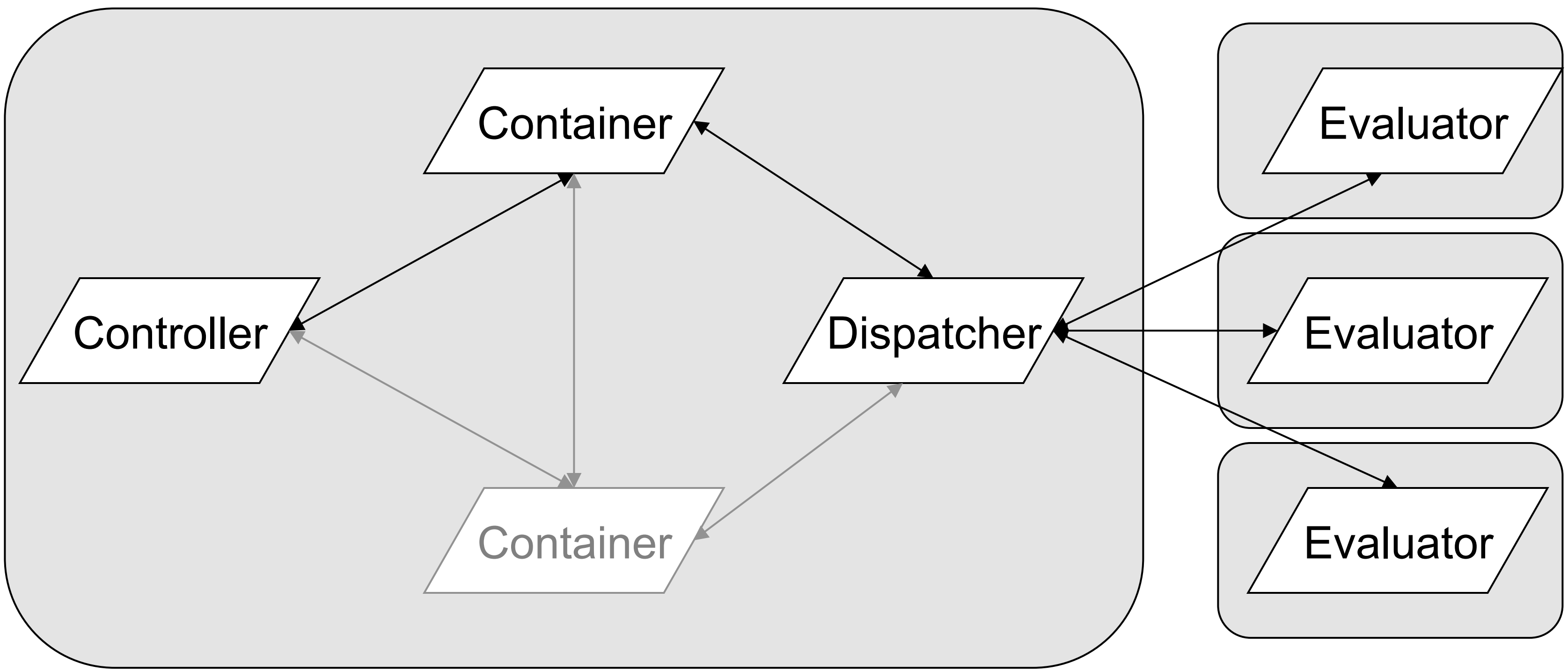
Physical View

- Given N computers, M populations, what goes where?
  - Nature of fitness
  - Nature of operators
  - Nature of computers
  - Nature of network
- Galapagos is flexible enough to be deployed almost anywhere

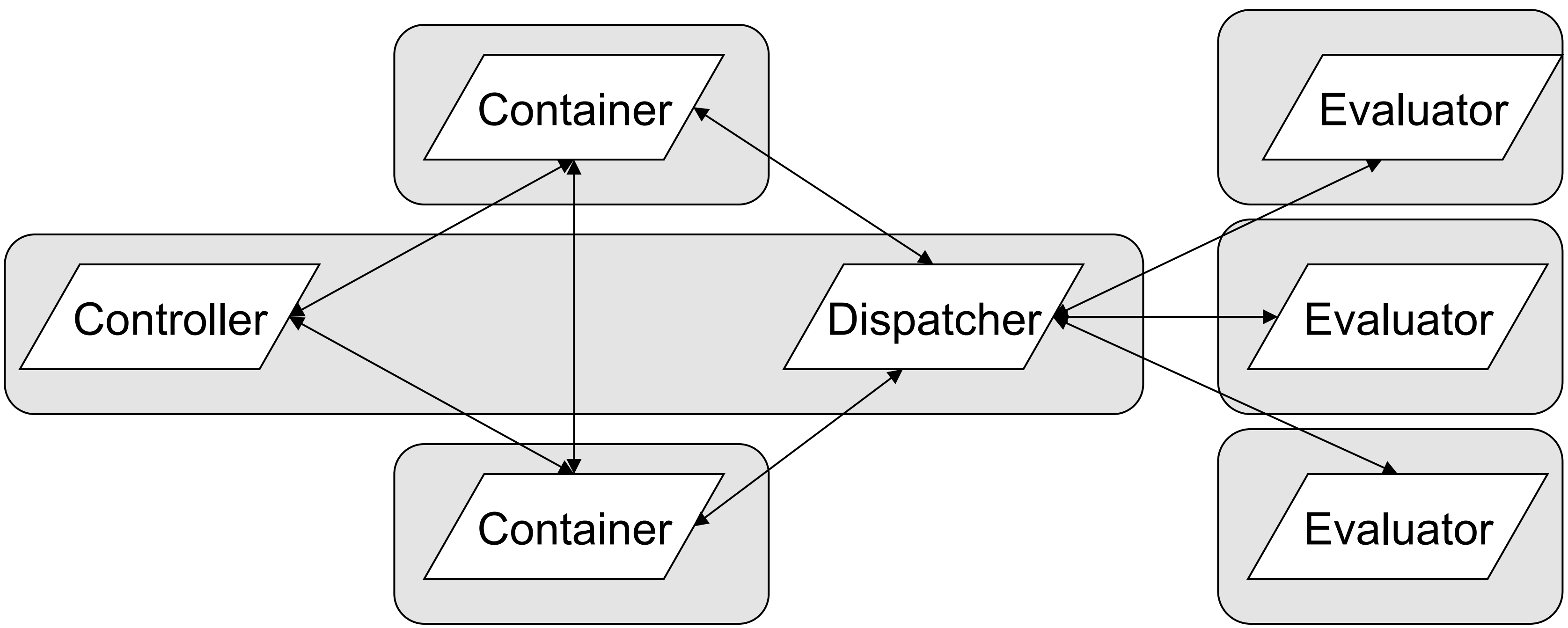
Physical Mappings



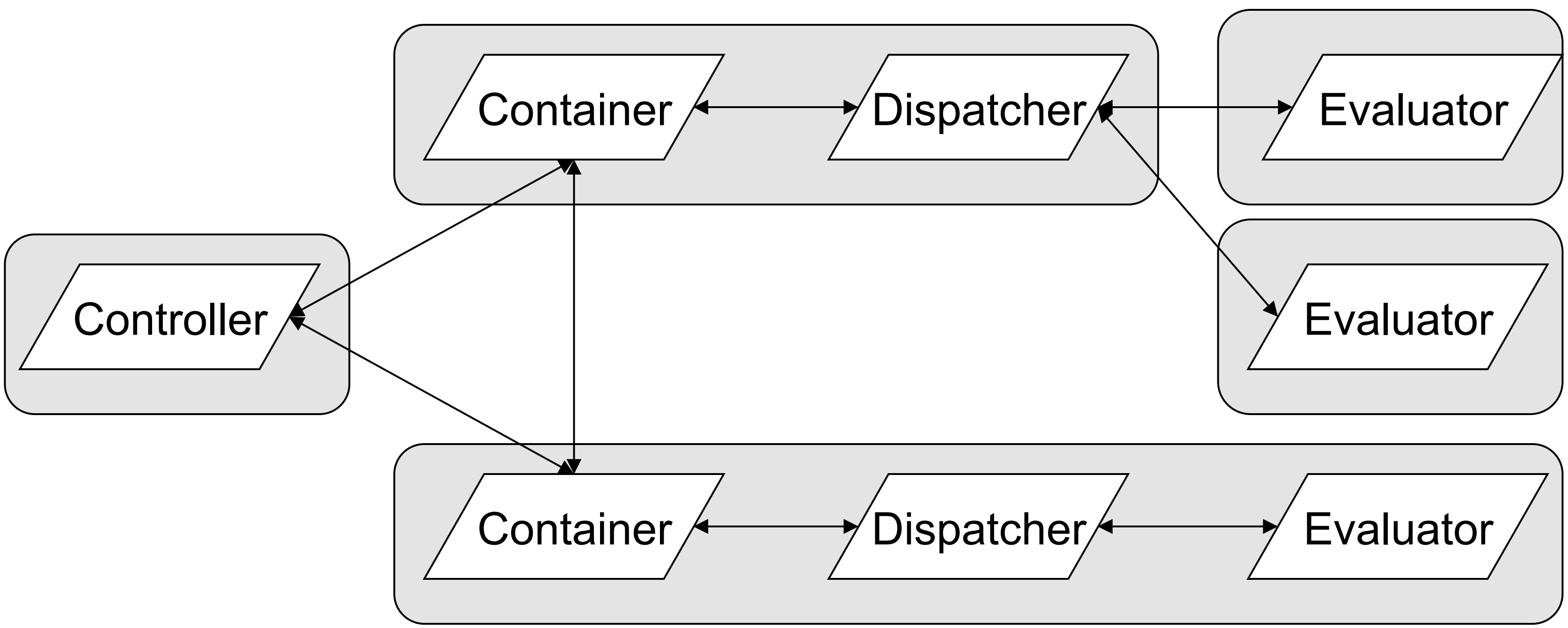
Physical Mappings



Physical Mappings



Physical Mappings



Summary

- Distributed PGA design is very complex:
  - Problem-domain specifics
  - GA type & population structure design
  - Available hardware
- Galapagos doesn’t provide easy answers but does provide a flexible common basis to work from

Questions?

nicolas@kruchten.com

this presentation is available at:

http://nicolas.kruchten.com/

Galapagos:

a generic distributed parallel genetic algorithm development platform

Nicolas Kruchten

4th year Engineering Science

(Infrastructure Option)