

# Java Parallel Processing Framework



***An Open Source Alternative  
to Grid Computing***

# Agenda

- What is JPPF?
- Features at a glance
- JPPF Architecture
- J2EE Integration
- Administration and monitoring
- Roadmap

# What is JPPF?

- **General-Purpose Grid Toolkit**
  - Federate computing resources working together
  - Handle large computational applications
  - Handle data-intensive problems
- **A Java framework**
  - Ubiquitous programming platform
  - OS and hardware independent
  - A platform for integration, extension, customization
- **An Open-Source Grid Environment**
  - Flexible licensing (Apache v2.0)
  - Source code guarantees transparency
  - Community-driven development process

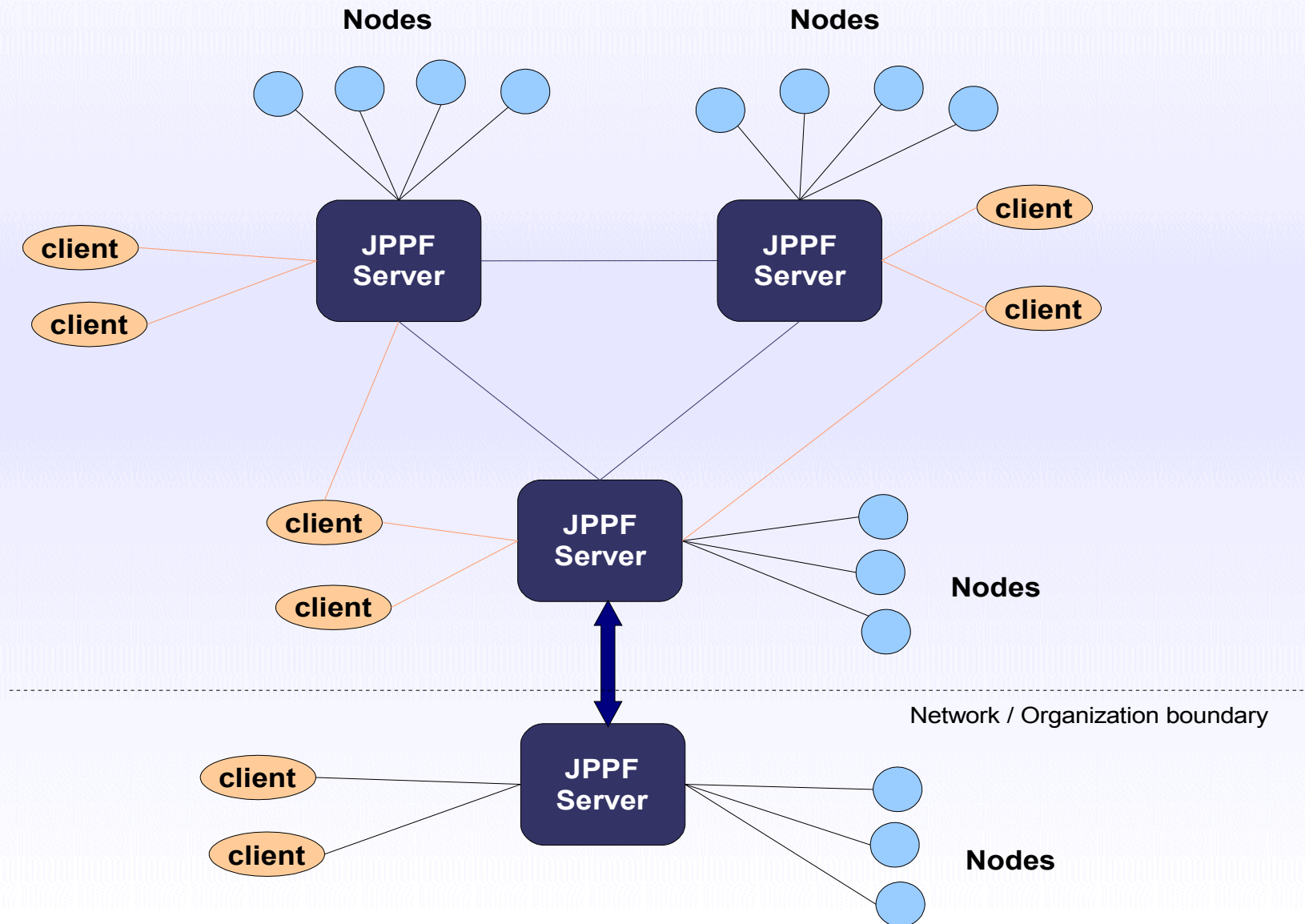
# Features

- Ease of use and deployment
  - write once, deploy once, execute everywhere
- Scales from small to large networks
- Multiple deployment options
  - standalone, OS services, JPPF@Home
- Configurable security
- Easy programming model
  - abstracts the complexity of distributed parallel programming
- Flexibility of integration

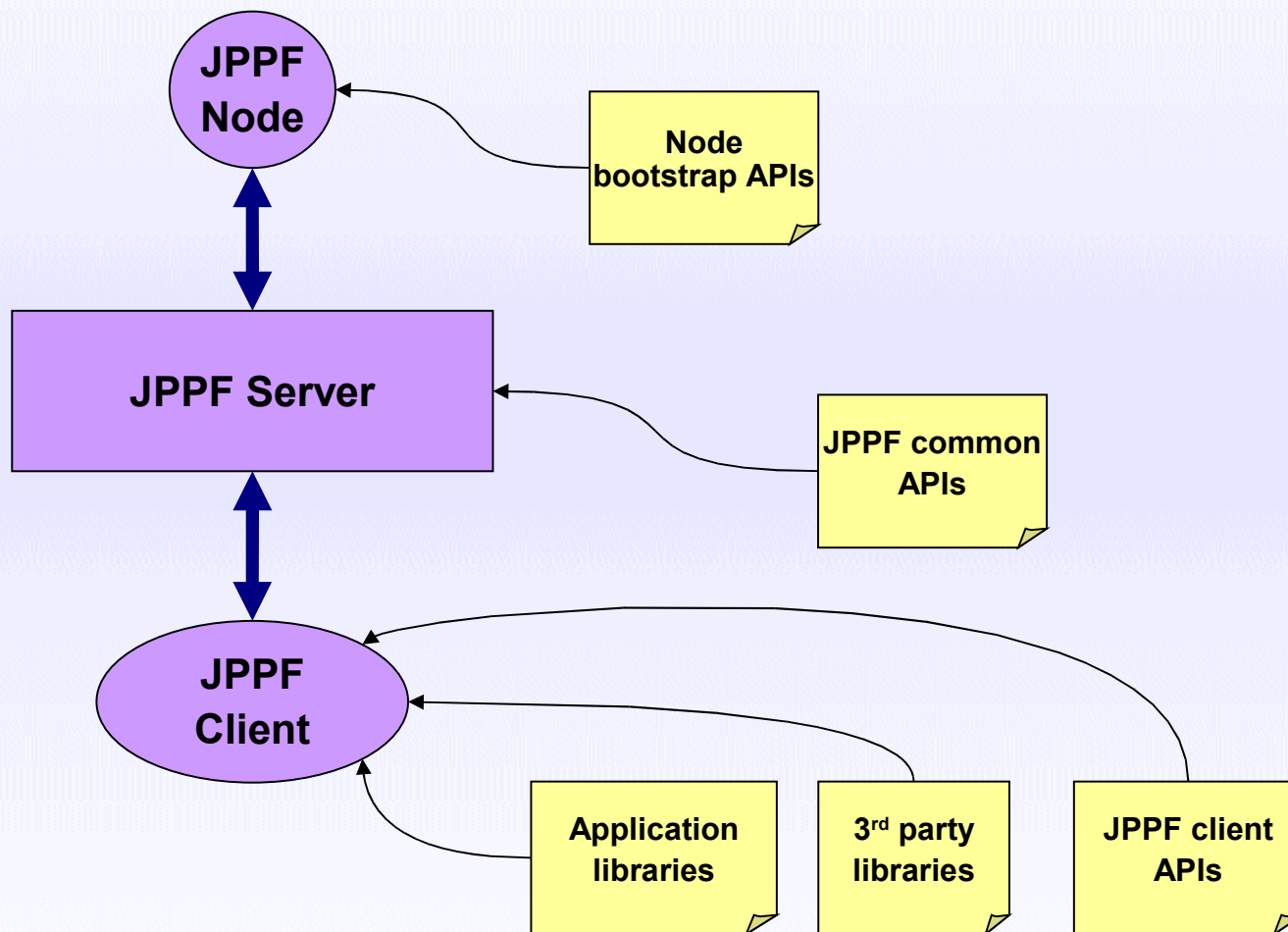
# Features

- Scalable distributed communication model
  - Consistent protocol between components
  - Adaptive load balancing
  - Optimized bandwidth usage
- Robustness
  - Built-in failover
  - Finely tunable recovery behavior
  - No single point of failure
- High performance
  - Small framework overhead
  - Asynchronous, non-blocking I/O
  - Continuous, feedback driven performance optimization

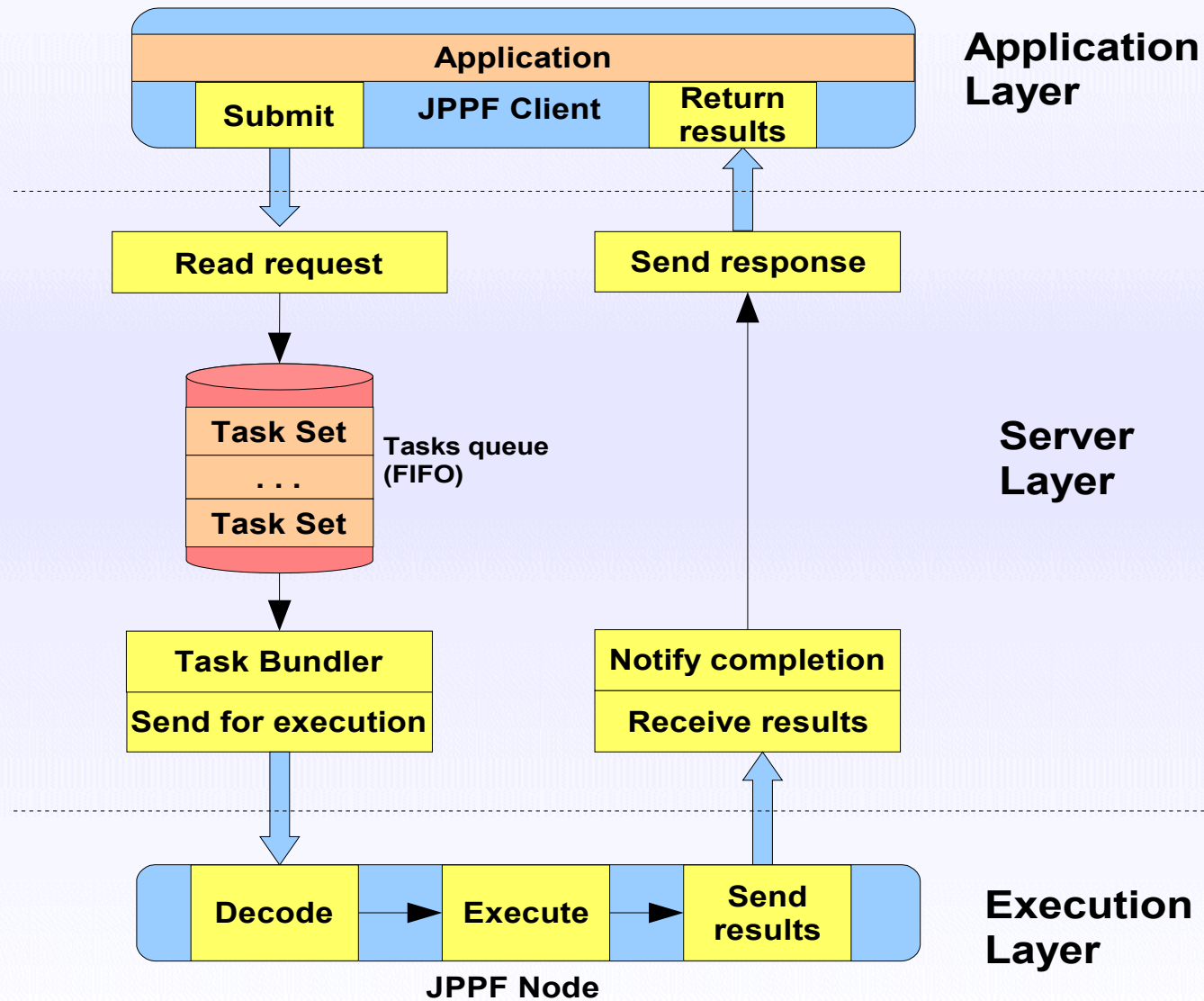
# Redundant Topology



# Deploy Once, Execute Everywhere



# Sound engineering



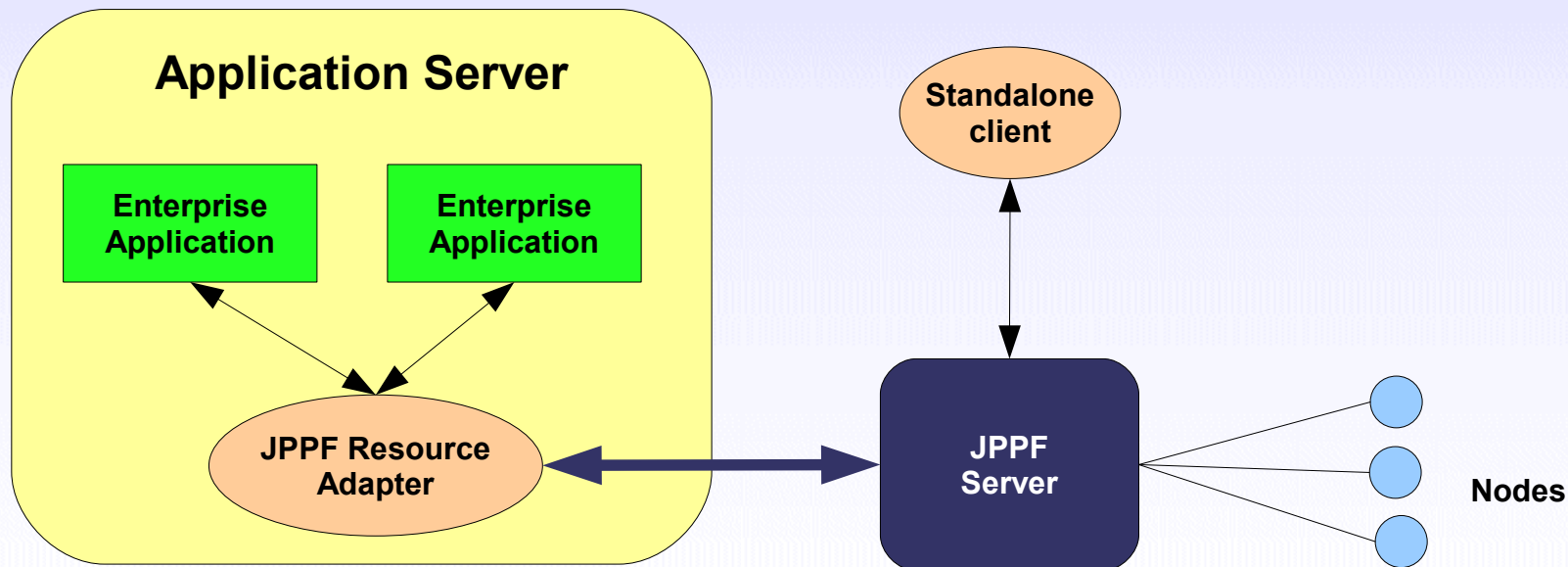


# Using idle CPUs



# J2EE Integration

- Works with leading application servers
  - JBoss, Websphere, Weblogic, OC4J, SunAS
- Leverages JCA 1.5
  - seamless J2EE integration
  - vendor independance
- Works as a JPPF client



# Monitoring and Management

- Fully customizable graphical interface
- Multi-Server, multi-node administration
- Fine Grained administration
- Customizable charts
- Internationalization support

The screenshot shows the JPPF Monitoring and Administration Tool interface. The server connection is set to 'driver2 : ACTIVE'. The 'Node Data' tab is selected. The interface includes a 'Toggle automatic structure refresh' checkbox (checked), 'Expand all', 'Collapse all', and 'Refresh tree' buttons. Below is a table with the following data:

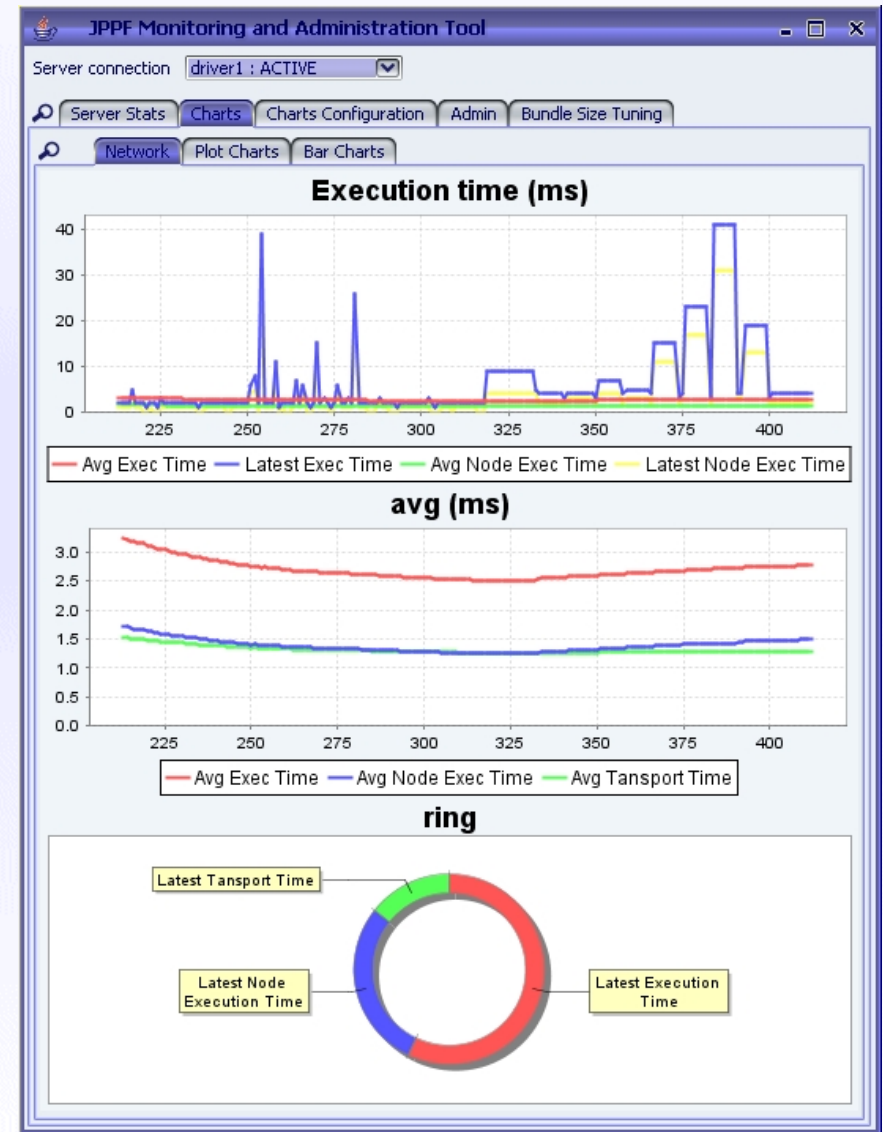
Node	Node Status	Exec status	Tasks Executed	Task Status
driver2				
localhost:12004	END_CONNECT	TASK_EXECUTED	25000	end exec
driver1				
localhost:12002	END_CONNECT	TASK_EXECUTED	24741	end exec
localhost:12001	END_CONNECT	TASK_EXECUTED	25259	end exec

The screenshot shows the JPPF Monitoring and Administration Tool interface with the server connection set to 'driver1 : ACTIVE'. The 'Node Data' tab is selected. The interface includes a 'Toggle automatic structure refresh' checkbox (checked), 'Expand all', 'Collapse all', and 'Refresh tree' buttons. Below is a table with the following data:

Node	Node Status	Exec status
driver1		
192.168.0.3:12001	END_CONNECT	START_EXEC

A context menu is open over the table, showing 'Cancel' and 'Restart' options. The 'Restart' option is expanded, showing a list of task IDs: Task id 3, Task id 2, Task id 0, and Task id 1.

# Monitoring and Management



# Roadmap

## Project vision

- ETL integration
- Business Intelligence back-end solution
- Globus & GigaSpaces integration
- Web Services integration
- Framework management automation
- Pluggable services
- Business Rule Engine integration

# Thank You

