

# Mercury: A reflective middleware for automatic parallelization of Bags-of-Tasks

João Nuno Silva, Luís Veiga, Paulo Ferreira

technology  
from seed



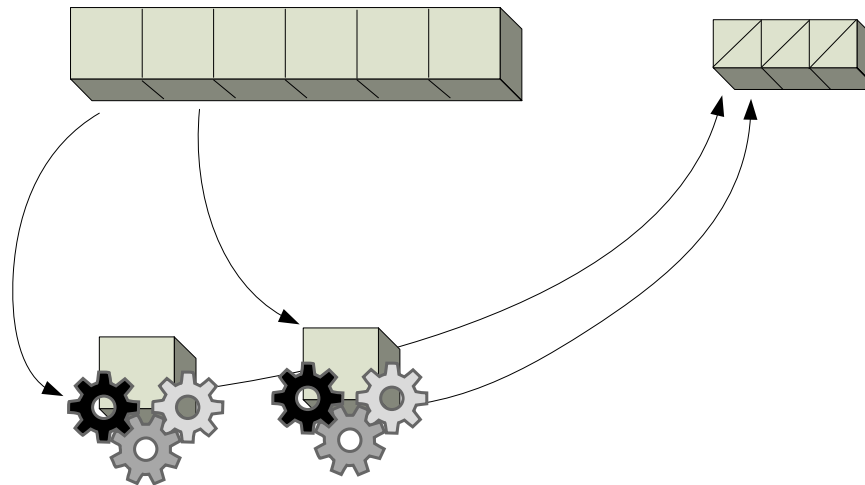
# Motivation



technology  
from seed

- Recent developments on parallel, large scale and distributed computing
  - Allow a broader set of resources
  - Allow an easier access to resources
- No increase in user community
  - Not significant development of parallel programming tools
  - Still hard to parallelize code

- Bag-of-tasks
  - Embarrassingly parallel problems
  - Fully independent tasks
  - Easy to deploy on parallel environments



- Programming Bag-of-tasks
  - Need for application modification
    - To transform serial version
  - Use of specialized libraries or middleware
    - Parallel tasks invocation
    - Data transmission
    - Threads, MPI, Map-Reduce

# Programming BoT

technology  
from seed



```
for i in range (1, 1000) :  
    input = getTaskInput(i)  
    obj[i] =  
        processClass()  
    obj[i].processData(input)
```

```
for i in range (1, 1000) :  
    result[i] = obj[i].getResult()  
process(result)
```

# Programming BoT

technology  
from seed



```
for i in range (1, 1000) :  
    input = getTaskInput(i)  
    obj[i] =  
        createParallelObject(processClass())  
        parallelExec(obj[i].processData(input))
```

```
for i in range (1, 1000) :  
    result[i] = obj[i].getResult()  
process(result)
```

# Programming BoT

technology  
from seed



```
for i in range (1, 1000) :
    input = getTaskInput(i)
    obj[i] =
        createParalelObject(processClass())
        parallelExec(obj[i].processData(input))
barrier()
for i in range (1, 1000) :
    result[i] = getResults(obj[i])
process(outputResult)
```

# Shortcomings



technology  
from seed

- Use of specialized API
  - Need to learn new API
  - Need to adapt the original application
    - Different data / code organization
  - Tied to a particular platform
- Extra complexity
  - Error prone
  - May not justify gains

# Ideal Solution



technology  
from seed

- Automatic / transparent transformation
- Aspects / Application annotation
  - Specialized compilers
  - Specialized VM
  - Not automatic nor transparent

# Proposed Solution



technology  
from seed

- Mercury
  - Run-time parallelization and distribution
  - Transparent transformation of serial code
  - External declaration of parallel methods
- Regular VM
- Only one code base
- Multiple execution environments

- User supplies
  - Application Code
  - List of the “parallel” methods
- Tasks processing methods
  - No interaction with external objects
  - Executed serially / concurrently

# Mercury Challenges

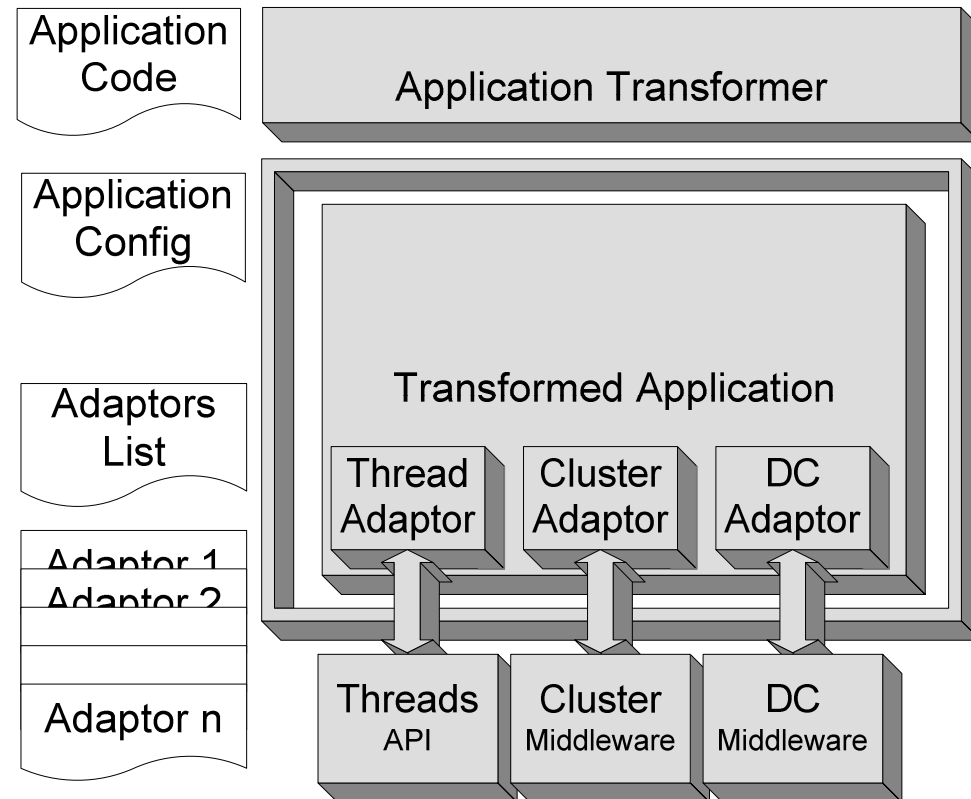


technology  
from seed

- Interception
  - Class loading
  - Object creation
- Methods synchronization
- Distribution of work

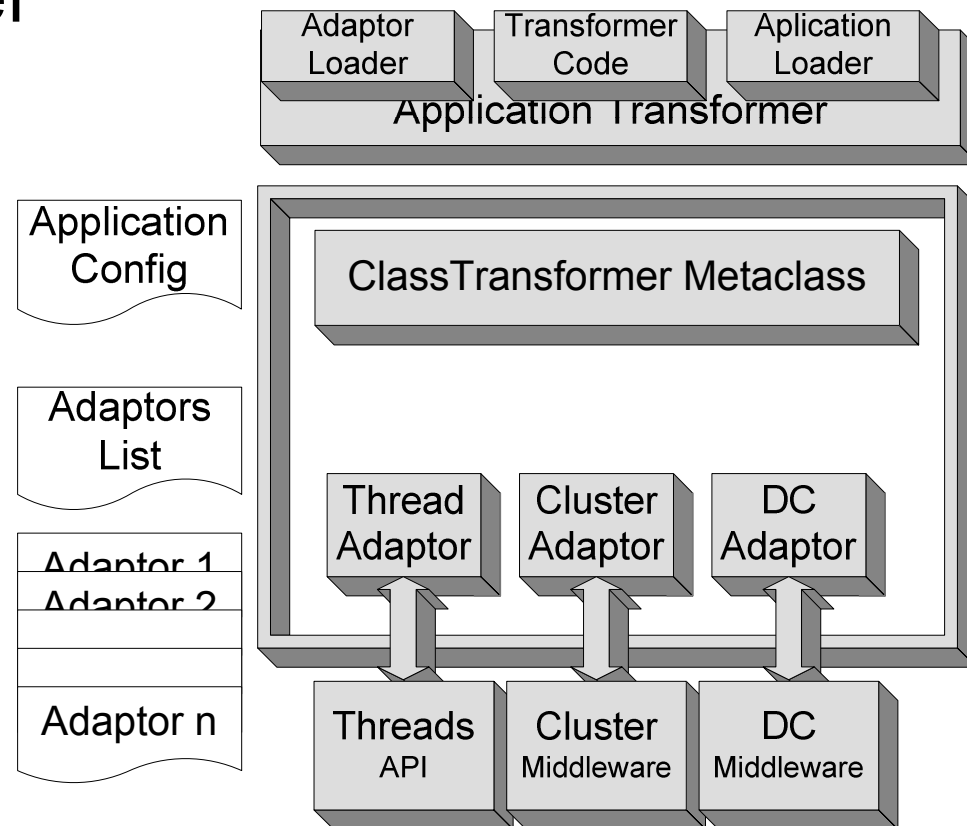
# Mercury Architecture

- Unmodified VM
- Adaptors
  - To interact with execution environment
- Application Transformer
  - Loaded on startup
- Application Code
- Application Configuration
  - Stating parallel methods



# Mercury Startup

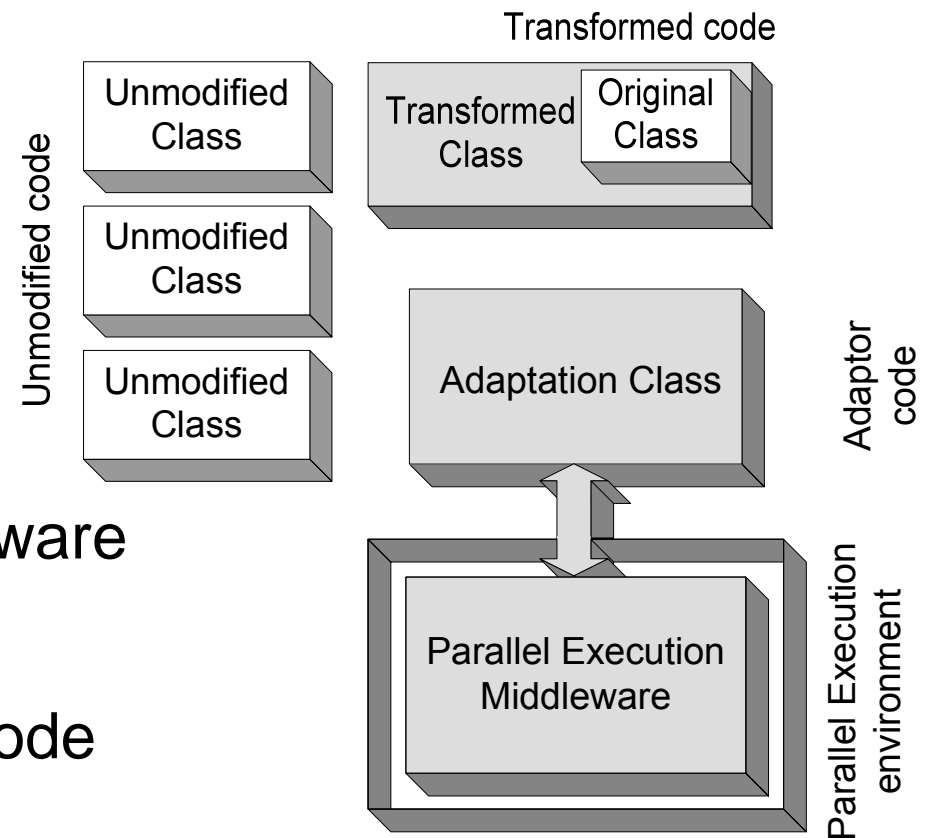
- ApplicationTransformer
  - Loaded on VM startup
  - Reads configuration
    - “Parallel” classes
    - Adaptors information
  - Creates Adaptors
    - For each available infrastructure
  - Installs a Metaclass



- Metaclass intercepts all class loading
  - Verifies “parallel” methods
  - Creates original unmodified class
  - Creates Transformed Class
    - Encasing original class code
    - Has the name of the original class
    - Will act as original class

# Class Hierarchy

- Unmodified code
  - interact with Original Classes/objects
- Adaptation Classes
  - Parallel Execution Middleware
- Transformed Classes
  - Contains Original Class code
  - Intercept object creations
  - Object factories



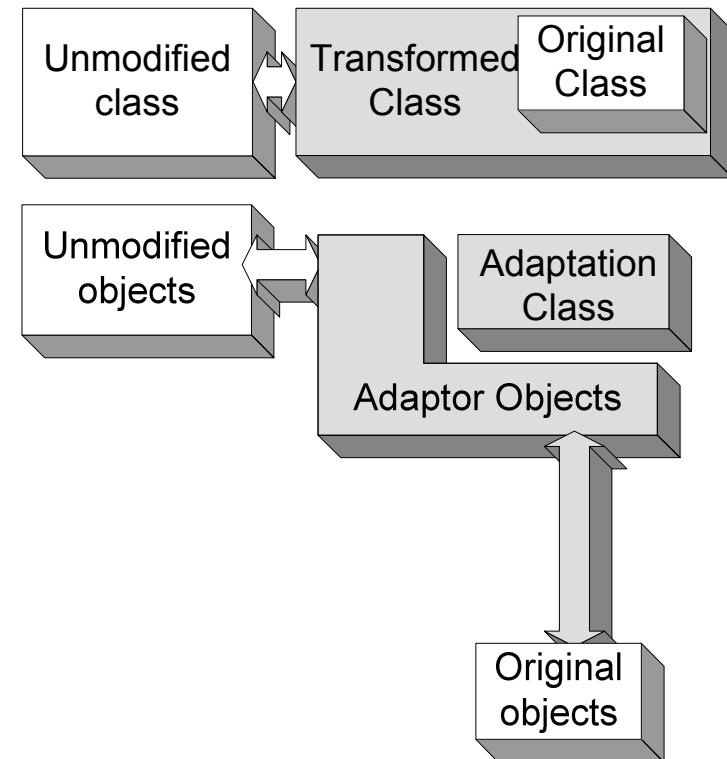
# Object Creation



technology  
from seed

- “Original class” instantiation
  - Intercepted by the Transformed Class
- Transformed class
  - Acts as an object factory
  - Selects execution environment
  - Creates instances of suitable adaptation class

- Adaptor Objects
  - Instances of Adaptation Class
  - Created by the Transformed Class
- Original Objects
  - Created by the Transformed Class
- Intercept calls
  - from unmodified objects
  - to originals objects

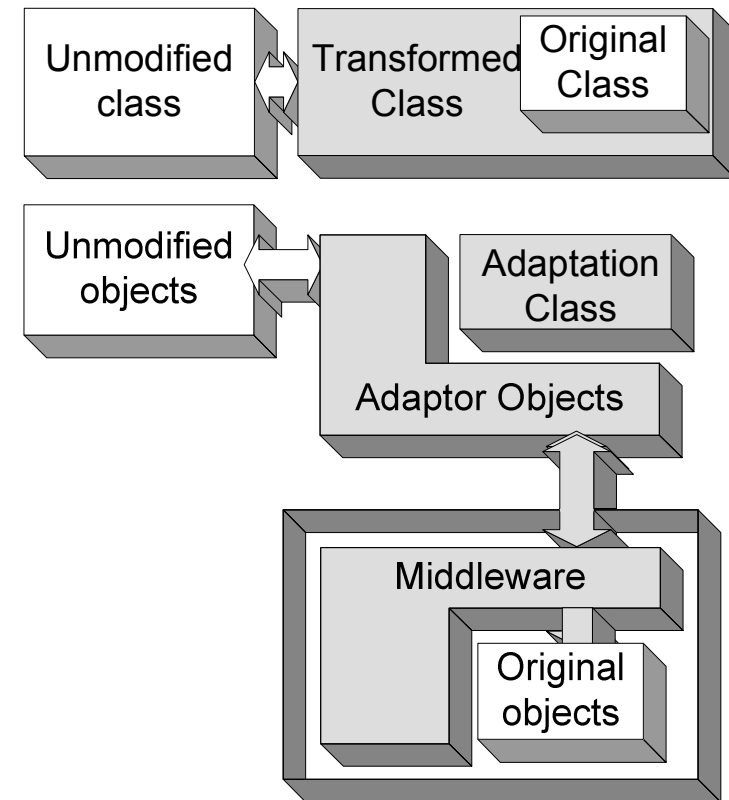


# Parallel Execution

technology  
from seed



- “Ordinary” calls
  - routed to the original objects
  - Synchronized
- “Parallel” calls
  - routed to the original objects
  - executed on a different thread
- Middleware provides
  - Parallelization / Distribution
  - Synchronization
  - Threads API
  - Remote/Mobile Objects



# Mercury Implementation



technology  
from seed

- Python
  - Widely used on parallel problems solving
    - Physics, Biology
    - Efficient specialized libraries
  - Provides Metaclass programming
    - without VM modification
  - Has Run time object/class modification
    - without code recompilation

# Mercury Evaluation



technology  
from seed

- Qualitative evaluation
  - On multiple environments
    - Multicore / cluster
- Evaluation of incurred overhead
  - From serial version to parallel
- Evaluation of speedups

# Mercury

## Qualitative evaluation



technology  
from seed

- Automatic parallelization of
  - function integration using Monte-Carlo
  - Executed on a Multicore
    - Thread API
  - Executed on a cluster
    - Mobile code middleware
- Only required
  - Transformation from imperative to OO

# Mercury

## Transformation overhead



technology  
from seed

- Integration of a function
  - Monte-Carlo method
  - 50 million iteration / 60 s serial version
- Imperative to OO (16 tasks)
  - **1.9 s overhead** (3.3% increase)
- Inside mercury (1 CPU)
  - **2.87s overhead** (4.7% increase)

# Mercury

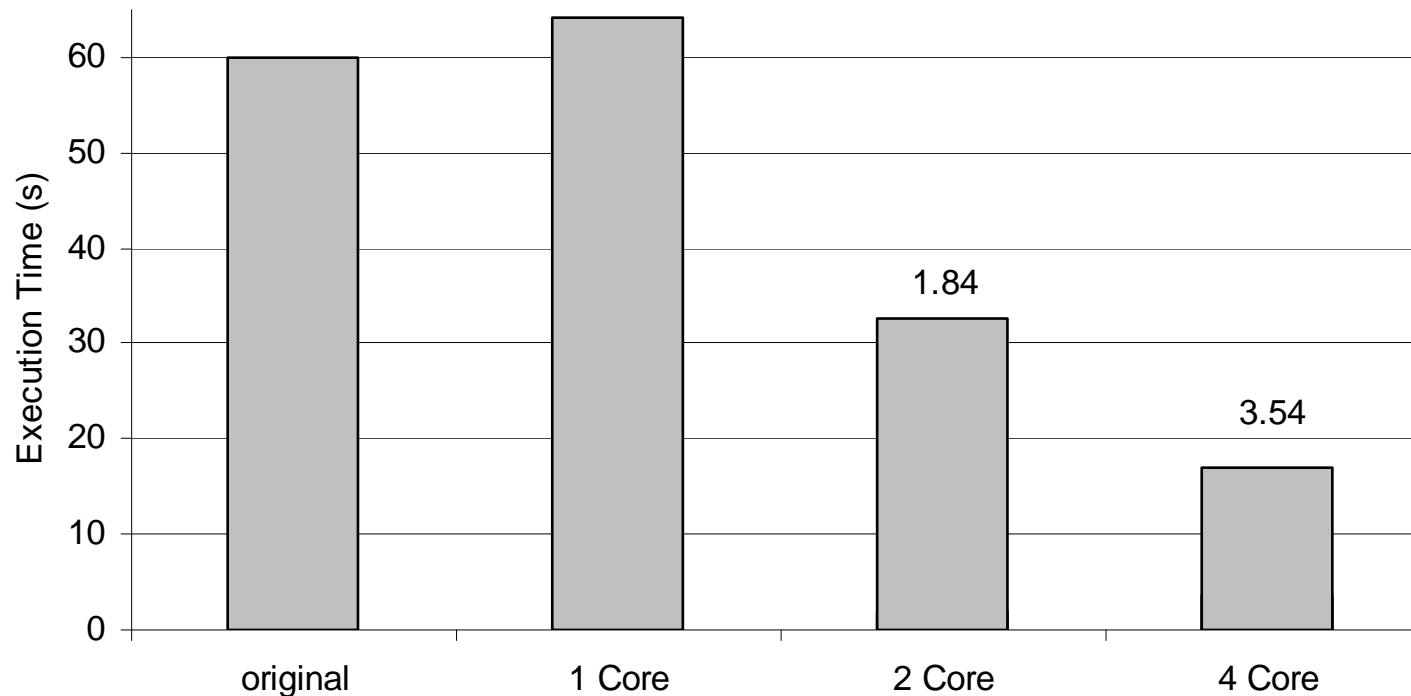
## Transformation speedups



technology  
from seed

- Integration of a function

- Intel Core 2 Quad @2.40GH (4 core)
- 16 tasks



# Conclusions



technology  
from seed

- Automatic and transparent parallelization tool
- Requiring minimal user intervention
- Adding minimal overhead
  
- Usable on multiple problems
  - BoT
  - Master / Slave (with barriers)
  - Simple workflows
- Independent from underlying infrastructure

Questions?  
[www.gsd.inesc-id.pt/~jnos](http://www.gsd.inesc-id.pt/~jnos)

technology  
from seed

