

**UNIVERSITY OF PATRAS  
COMPUTER ENGINEERING & INFORMATICS DEPARTMENT  
High Performance Computing Laboratory (HPCLab)**



**Parallel and Distributed Systems Group**  
<http://pdsgroup.ceid.upatras.gr>

Eleftherios Polychronopoulos  
Constantinos Karantasis (PhD)  
Anastasios Tsoukas (MSc)  
Nasos Antoniou (MSc)  
Alexandros Katsonis (MSc)

# Current Ongoing Research

## □ Runtime Systems & Programming Models

Pleiad: A Cross-Environment Middleware supporting efficient multithreading on top of distributed resources

## □ Parallelization of Scientific Applications

Fluid Dynamics Simulation (Dept. Mech. & Aerospace Eng.)

Modeling of nano-Carbene Tubes (Dep. of Chemistry)

## □ Auto Tuning Libraries (ATL)

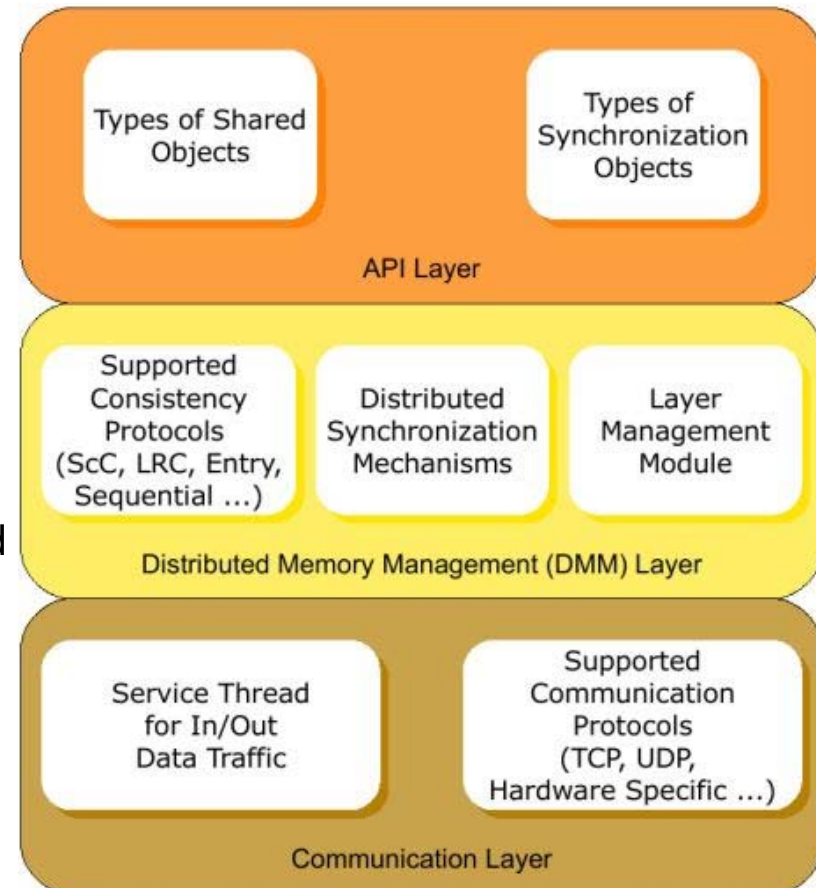
Developing a Parallel Shorting ATL

# RTS & Programming Models - **Pleiad**

- ❑ Origin: Old Research Idea - Software Distributed Shared Memory
- ❑ Incorporation of new Ideas:
  - ❑ Executes on multicore architectures
  - ❑ Supports execution on top of heterogeneous, non-uniform resources
  - ❑ Aims to be highly adaptable just before or during runtime
  - ❑ It is based on the latest Java platform (version  $\geq 5$ )
  - ❑ Library level implementation
  - ❑ Uses bytecode instrumentation to enforce tuning during runtime (data placement/thread migration)
  - ❑ Layered and modular architecture
  - ❑ Every distinct mechanism communicates with the others using well defined interfaces in a way that the implementation of every mechanism is not bounded to its interface. This makes feasible the interchange of implementations according to the needs of the application even during runtime.

# Plead - Architecture

- ❑ Organization on layers which include several mechanisms organized on extendable modules.
- ❑ For every mechanism one or more implementations are available.
- ❑ Network layer (currently based on TCP/IP sockets), incorporates a service thread, that manages the incoming / outgoing traffic asynchronously.
- ❑ Memory Management Layer, includes relaxed consistency models, invalidation/update policies and synchronization constructs.
- ❑ The API does not impose any extensions to the Java language specification and it is as easy to use as a Java collection.
- ❑ Provides shared abstraction for both objects and arrays.



# Pleiad Data Placement - Example use

---

```
public class Paradigm {
    ...
    public void deploy(String[] args) {
        SharedObject x =
            new SharedObject(new Integer(0));

        int chunk = 256;
        SharedArray v =
            new SharedArray(1600, chunk);

        LockObject mutex = new LockObject();
        BarrierObject bar =
            new BarrierObject(num_threads);

        for(int i = 0; i < num_threads; i++)
            Pleiad.setupThreads(args);
    }
    ...
}
```

---

---

```
public class ParadigmThread extends PleiadThread {
    public void run() {
        ...
        istart = size * id;
        iend = size * (id + 1);
        for(int i = istart; i < iend; i++){
            num = compute(v.get(i));

            mutex.lock();
            sum = x.get();
            x.set(sum + num);
            mutex.unlock();
            bar.wait();
        }
        ...
    }
}
```

---

# Pleiad On going Research

Currently working on:

- Transactional memory and the support of transactional execution in the context of Pleiad.
- Extending Pleiad to different types of computational resources. Including mobile and embedded devices through the integration of J2ME and Java Real-Time VMs.
- Integration of Pleiad with existing-available grid software.

# Parallelization of **Scientific Applications**

## □ Fluid Dynamics Simulation

**In conjunction with the Department of Mechanical & Aerospace Engineering  
(Professor John Aikaterinaris)**

We have two applications simulating Fluid Dynamics.

One coded in Fortran 90 and one in C.

The target is to Optimize and Parallelize both applications with:  
MPI, OpenMP, MPI & OpenMP, OpenMP & CUDA

## □ Modeling of nano-Carbene Tubes

**In conjunction with the Department of Chemistry  
(Professor Theodore Christopoulos)**

We have the program modeling

# Auto Tuning Libraries - Parallel Shorting

- ❑ The development of an Auto Tuning Library for parallel shorting

(In conjunction with the Group of David Padua – UIUC)

- ❑ Enhancing the library with existing algorithms
- ❑ Incorporating auto tuning mechanisms
- ❑ Parallelizing the library with:
  - OpenMP & MPI
  - CUDA

# Thank you

## Questions



Hipeac Computing Systems Week, Paris, 2008