Grid programming with components:
an advanced *COMP*onent platform
for an effective invisible grid

Load-Balancing for Multicast Interfaces

Matthieu Morel

University of Chile

Problem statement

1. Computational speedup through parallel resources

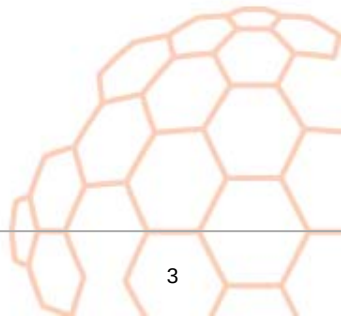
2. Paradigms
   - Tightly coupled (SPMD)
   - Divide & conquer
   - Service composition, Workflows
   - **Embarrassingly parallel** (some GridCOMP use cases)

- Efficiency depends on:
  - Modeling of the problem
  - Partitioning - size
  - Infrastructure

- Infrastructure is often :
  - Volatile
  - Heterogeneous

Philosophy of ProActive / GCM

- offer component-based programming
  - Separation functional - non functional
  - Inversion of control
  - Customization (controllers)
- provide common facilities
  - Deployment
  - Assembly
  - Communication

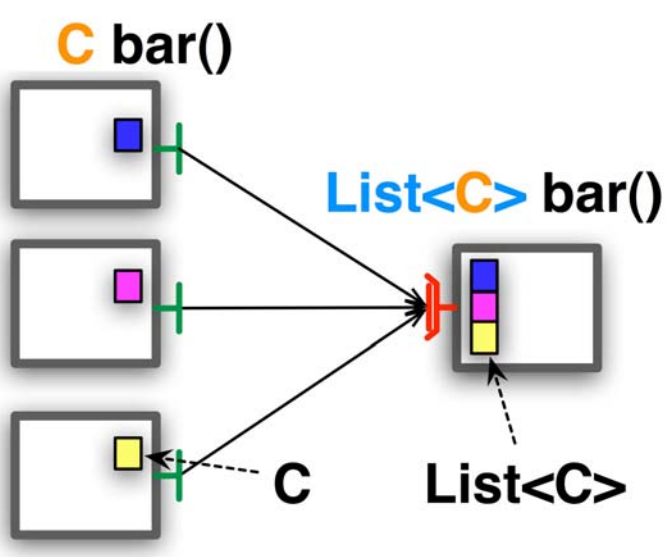


Solutions for embarrassingly parallel problems

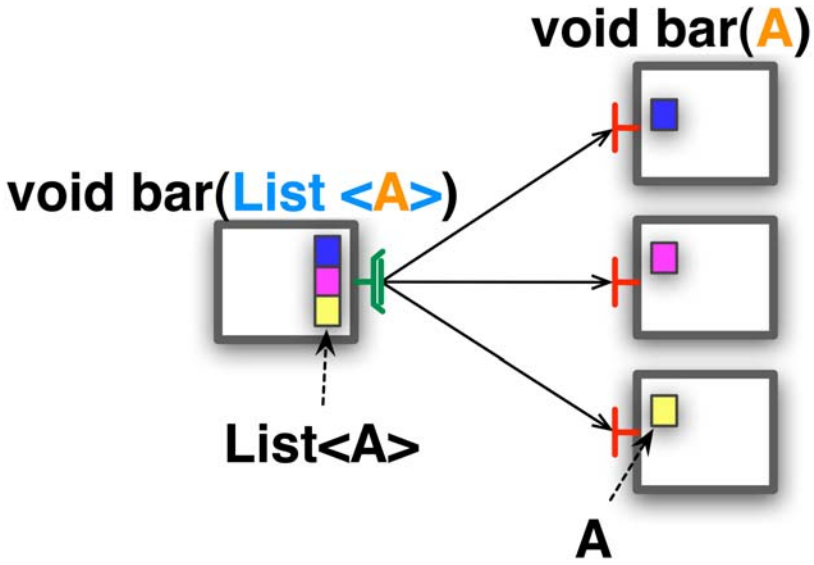
- Dedicated schedulers
  - Ex: *ourgrid* scheduler
    - ☹ Focus on task allocation
      Coarse grained tasks
- Alternative:
  - ☺ Focus on the problem
  - ☺ Structured assembly of components
  - ☺ Parameterized interactions

⇨ High-level programming **facilities**

Collective interfaces



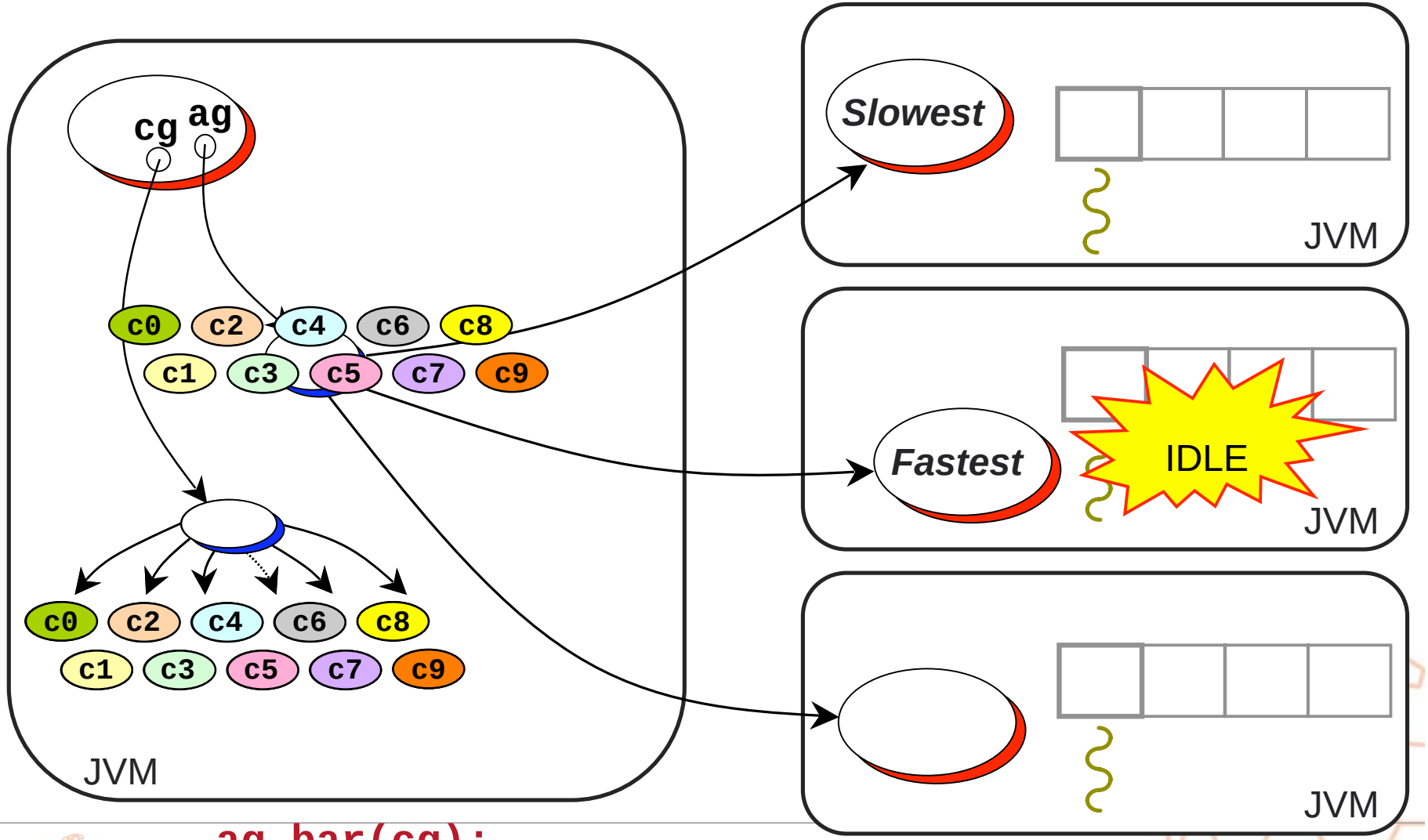
GATHERCAST



MULTICAST

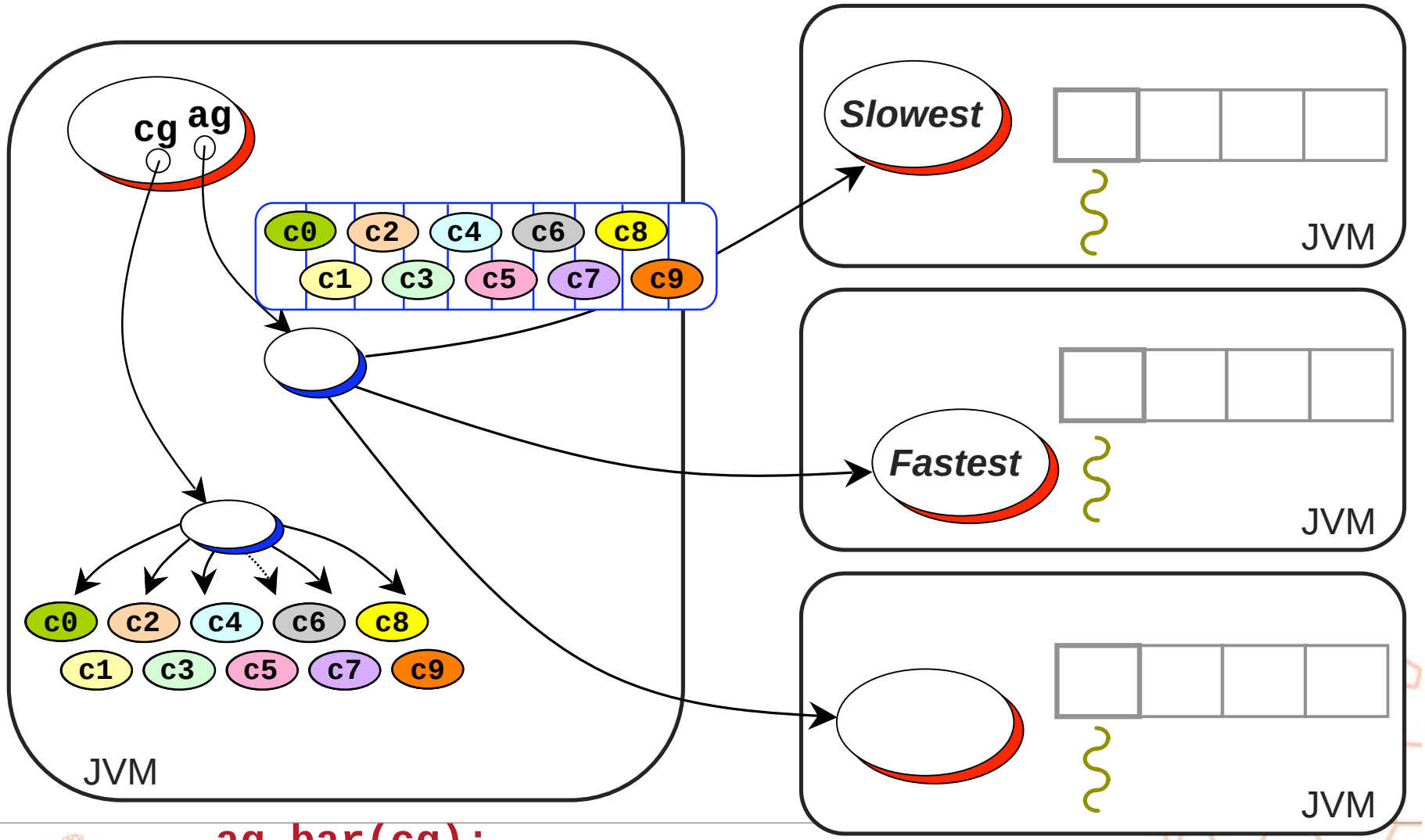
☹ Static dispatch

Static Dispatch Group



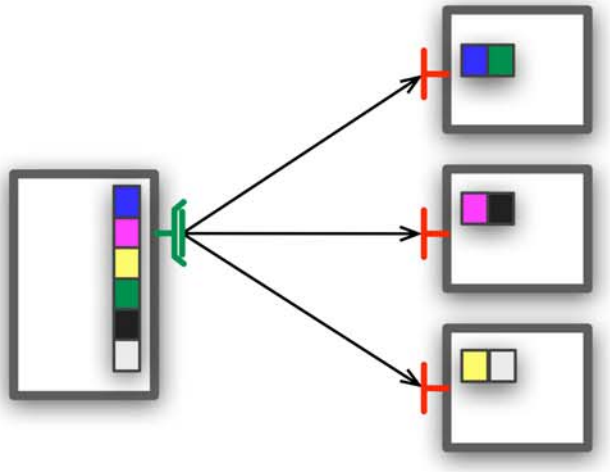

```
ag.bar(cg);
```

Dynamic Dispatch Group

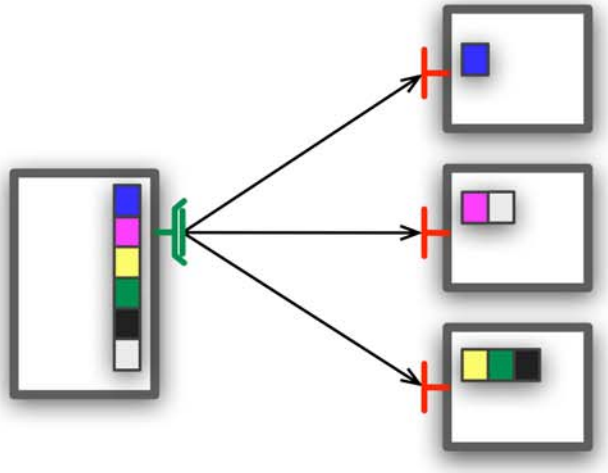



```
ag.bar(cg);
```

Dynamic Dispatch with Multicast Interfaces



uniform distribution



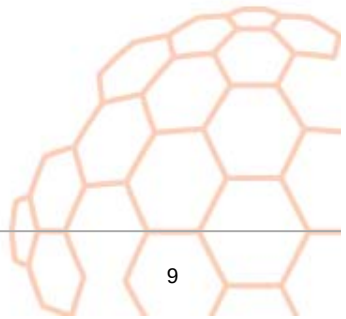
knowledge-based distribution

more efficient !

Principles

- Minimal scheduling facilities
  - Knowledge-based scheduling

    workload **+** network congestion
- GCM programming model
- Composition oriented vs task oriented
- Low-level integration in ProActive/GCM



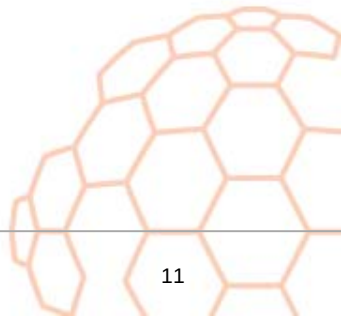
First achievements

- Load balances work units
- Compatible with POJO groups

- vs other frameworks:
  - Faster than ProActive's master-worker (low level)
  - Faster than *ourgrid* scheduler (fine grained tasks)
  - Comprehensive: splitting - scheduling - reduction
    (map-reduce / split-aggregate)

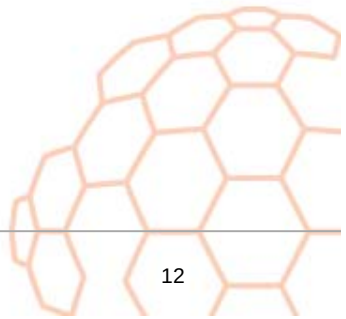
Impact on ProActive/GCM

- API preserved
- Extensions to Meta-Object Protocol
- Open implementation
- Integration to codebase: ProActive v4.0?



Side Contributions

- Bug fix for groups
  - “swallowed parameters” error : not all parameters distributed in some cases
- Relies on Java 5 concurrency features
  - More stable thus efficient for high loads



Future Work

- Finish integration (includes configuration spec)
- Use runtime load information
  - Aldinucci's work : tagging futures
- More standard dispatch modes
  - Random
  - Predictive CPU based?
- Unicast dispatch (probably short-term task)

Questions?

- applicability to adaptable farms?
  - ⇨ Parameterizable dispatch function
- suitability for GridCOMP use cases?
  - ⇨ Yes : simple mechanism

Contact:

Matthieu Morel mmorel@dcc.uchile.cl