Dynamic Parallel Objects for Metacomputing

Nguyen Tuan Anh, Pierre Kuonen

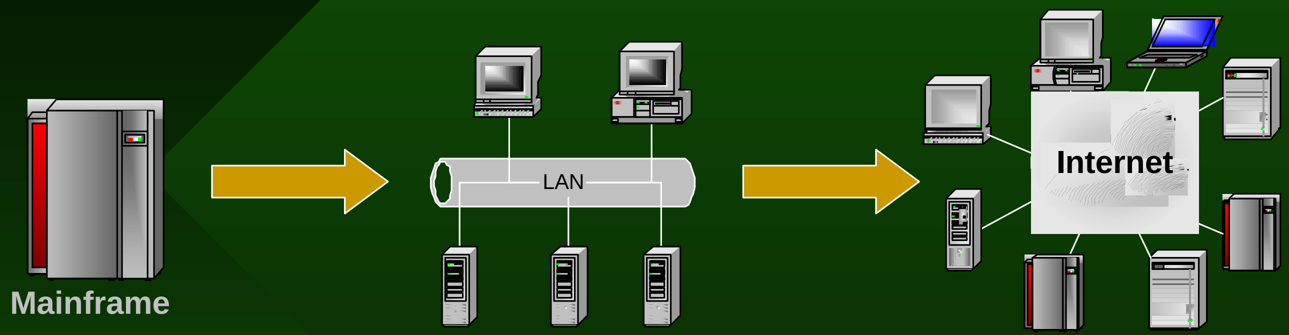
University of Applied Science Valais, Switzerland

10/31/2001

Outline

- Metacomputing and intensive high performance computing
- Dynamic parallel object model and object infrastructure
- Some related works: Legion, CORBA, COBRA
- Case study: pattern and defect detection system for textile manufacturing
- Conclusion

Metacomputing



- Large number of wide area distributed resources
- All resources are connected by the Internet, forming a virtual parallel computer
- Resources can be data storages, sensors, workstations, supercomputer, etc.

Intensive High Performance Computing Applications

- Strict time constraints
- Enormous computing power required
- Huge data processing
- Computation on demand
- Complex application structure with multiple level of parallelism

IHPC application on metacomputing environment

- Computational model should adapt to the dynamic state of the environment
- Efficient use of large pool of metacomputing resources
- Preserving the performance of the application
- Fault tolerance

Object-oriented parallelism

- Two approaches: method parallelism and object parallelism
- Method parallelism:
  - Method interface unchanged, parallelization inside method
  - Suitable for fine grain parallelism
  - Hard to implement on distributed environment
  - Breaking object oriented paradigm
- Object parallelism:
  - Dividing objects into small objects by data partitioning, function partitioning
  - Each object is an entity
  - Natural way of parallelism
  - Suitable for coarse to medium grain of parallelism

Dynamic Parallel Objects

- Object parallelism
- Parallel objects:
  - Located on different resources
  - Some operations in a parallel object can be called by other objects in parallel (or at least concurrently)
  - Operations on different parallel objects can be executed in parallel
  - The creation of parallel object is transparent to users
- Interaction between objects through object interfaces

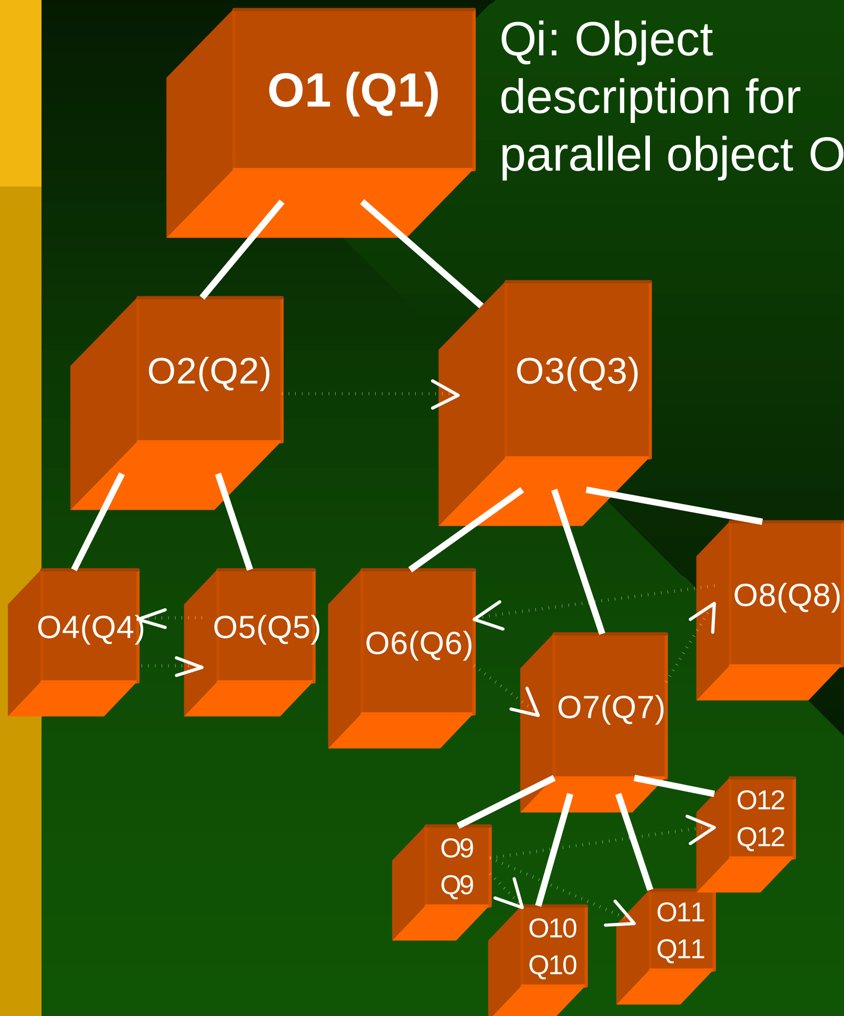
Object Description

- Each parallel object has a user-specified object description
- Describing the requirement of parallel objects
- Will be used as a guideline for allocating resource
- Can be expressed in terms of:
  - Maximum computational power (e.g. Mflops)
  - Communication with other parallel objects
  - Memory needed
  - Strict requirement or not

Dynamic Parallel Objects

- The problem to be solved: a parallel object
- Parallelization of a parallel object can produce other parallel objects
- Problem can be solved by:
  - Invoking operations on the object or
  - Replacing the problem object by its parallel objects and invoking operations on these objects
  - Operations on a parallel object can also be replaced by operations on its descendant parallel objects

Dynamic Parallel Object Model

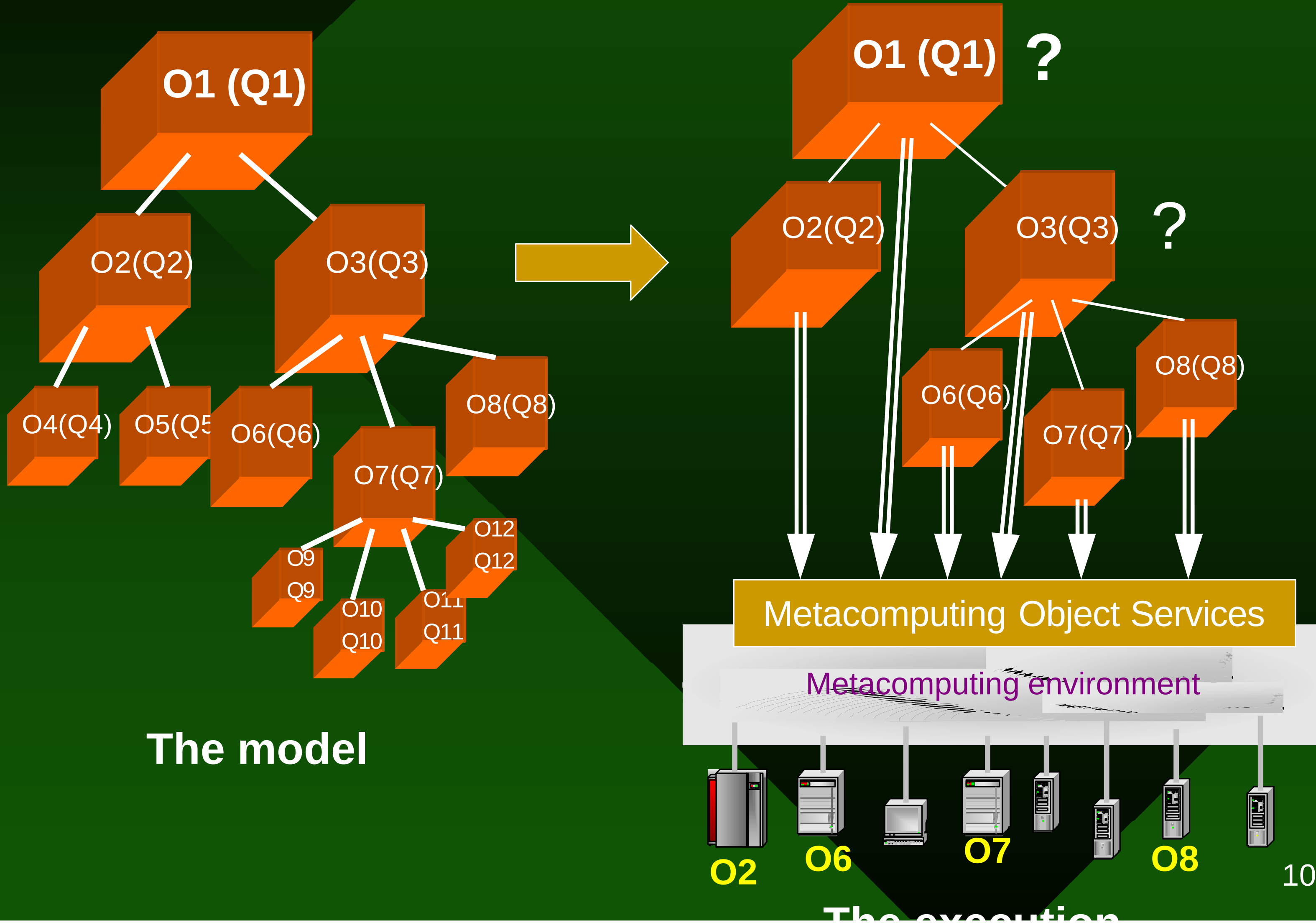


Qi: Object description for parallel object Oi

- Problem can be represented by objects:
  - O1 or
  - O2, O3 or
  - (O4, O5), O3 or
  - O2, (O6, O7, O8)

  …
  - (O4, O5), (O6, (O9, O10, O11, O12), O8)

Dynamic Parallel Object Model



Dynamic Parallel Objects

- Parallelism by:
  - Replacing a parallel object by its descendant parallel objects
  - Interaction between parallel objects through object interfaces and independent from their parents
  - Parallel invocation of different methods on different parallel objects
  - Parallel invocation of the same method by different objects (sharing parallel object)

Characteristic

- Parallelism model, object-oriented approach
- Time constraints: users specify the time they desire their problem to be solved
- Distributed parallel computing
- Computational resources do not need to know in advance
- Multi-level and dynamic parallelism
- The number of parallel objects is dynamic and only decided during the run time

Advantage

- Support IHPC on metacomputing environment
- Complex and multiple level of parallelism, from coarse to fine grain
- Suitable for metacomputing environment. The parallelism will be dynamically adapted to the current availability of resources
- Object oriented technology

Disadvantage

- The complexity of problem should be known or at least the user should have an educational guess
- Users have to decide all possible ways of parallelization

The object infrastructure

IHPC applications

Dynamic Parallel Objects

Object services

Invocation services | Allocation services | Object info services | Other object services

Metacomputing services

Object Services

- **Invocation services:** manage the marshalling, unmarshalling, transmitting of data and invoking methods of parallel object
- Allocation services: manage the resource discovery based on object description, transmitting of object code, and creating dynamically parallel objects on the remote resources
- Object information services: manage all information about parallel objects such as current locations of objects, location of object's code, etc.
- Other object services: reservation, security, monitoring, etc.

Related work: Legion

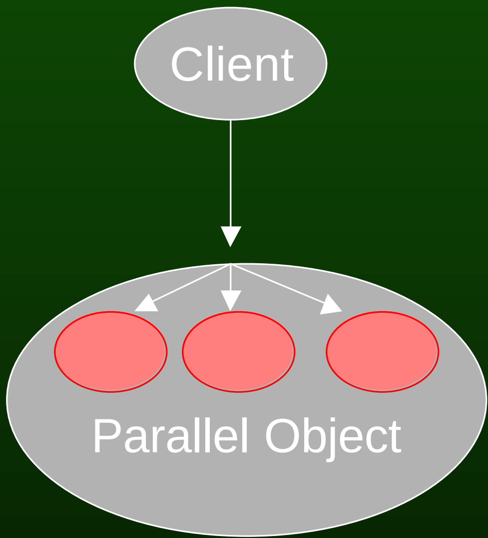
- A well-known project, providing an object-based infrastructure for meta/grid computing
- Service-centric approach
- Two states of objects: active (running) or passive (on the storage)
- Method calls are non-blocking. Parallelism through method invocation
- Data flow parallelism
- Lack of support for dynamic parallelism
- Object allocation based on requirement is not specified

Related work: CORBA

- A standard developed by Object Management Group
- Allowing remote method invocations based on Object Request Broker
- Targeted client-server applications
- Not designated for high performance parallel applications
- No parallelism model

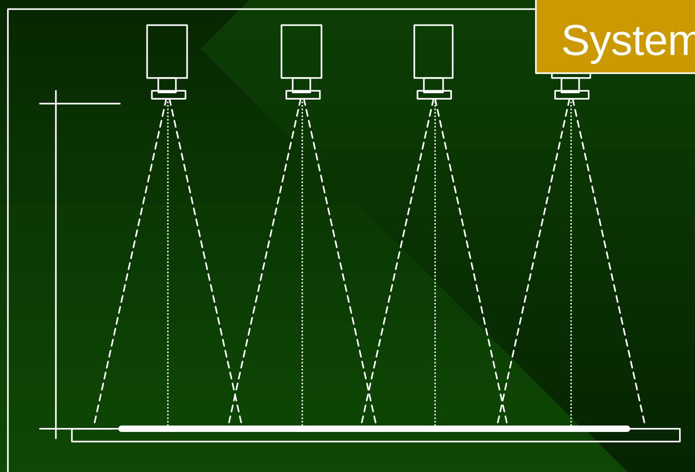
Related work: COBRA

- An extension of CORBA to support parallelism
- Encapsulate parallelism within parallel CORBA objects
- The concept of data parallel objects: parallelism mainly by data partitioning
- Limited level of parallelism
- Focus mainly on the interaction between a parallel object with other objects rather than the parallelism of objects themselves



Case study: pattern and defect detection system (PDDS)

Artificial Vision System (AVS) → images → **PDDS** → patterns → Nesting System



- **Technical info**
  - Textile width:0-1.7m, length:0-100m
  - Conveyor speed: 2-6m/min
  - Output (AVS): continuous image, 3.3MPixel/s

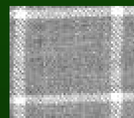
•Analyze images: find the positions of patterns and detect defective ones.

•Analysis speed: >3.3Mpixel/s.

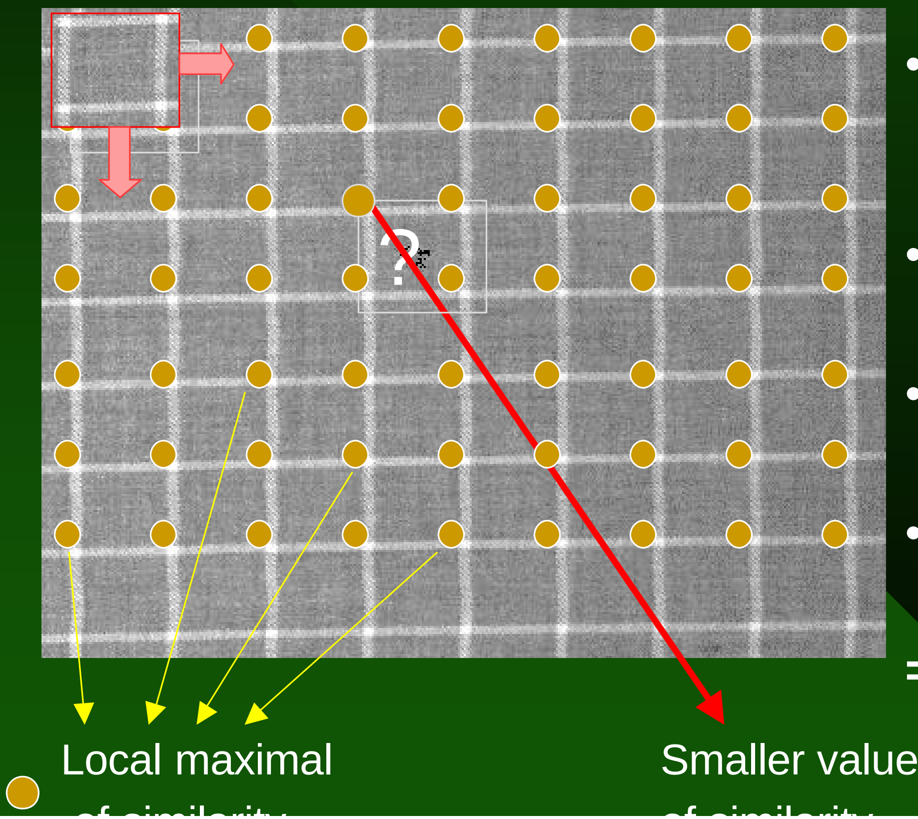
•Cutting tissue in order to minimize the amount of wasted material.

The PDDS Algorithm

•Pattern template:



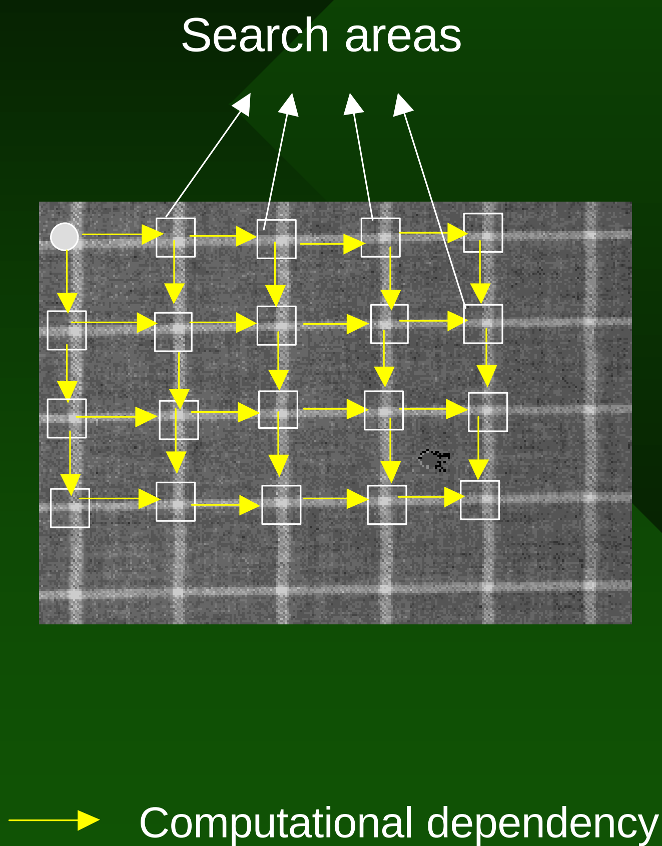
•Tissue image:



- Inputs: a pattern template and a tissue image
- Shift the template over the tissue image
- For each position, compute the similarity between the template and the sub-image
- Pattern position: local maximal of similarity
- Criterion for the similarity: mutual information
- Computing power needed: 226 GFlop/s!

=> Optimization and parallelization needed

The PDDS: Optimization

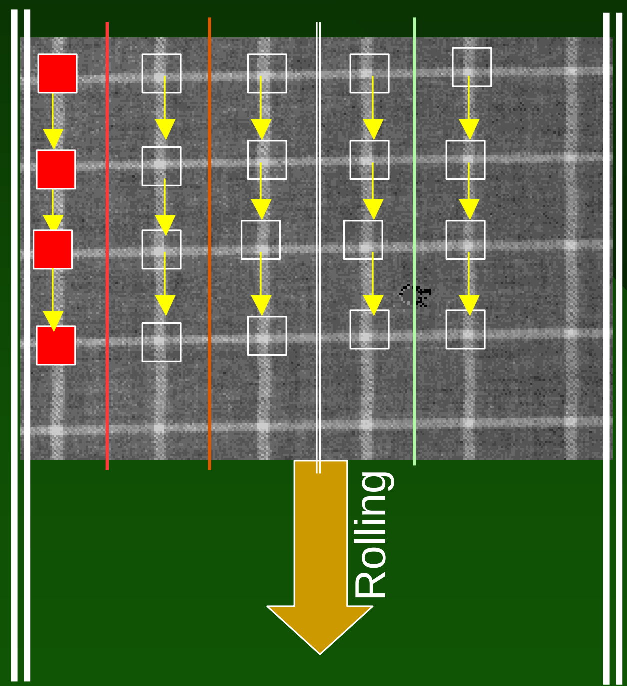


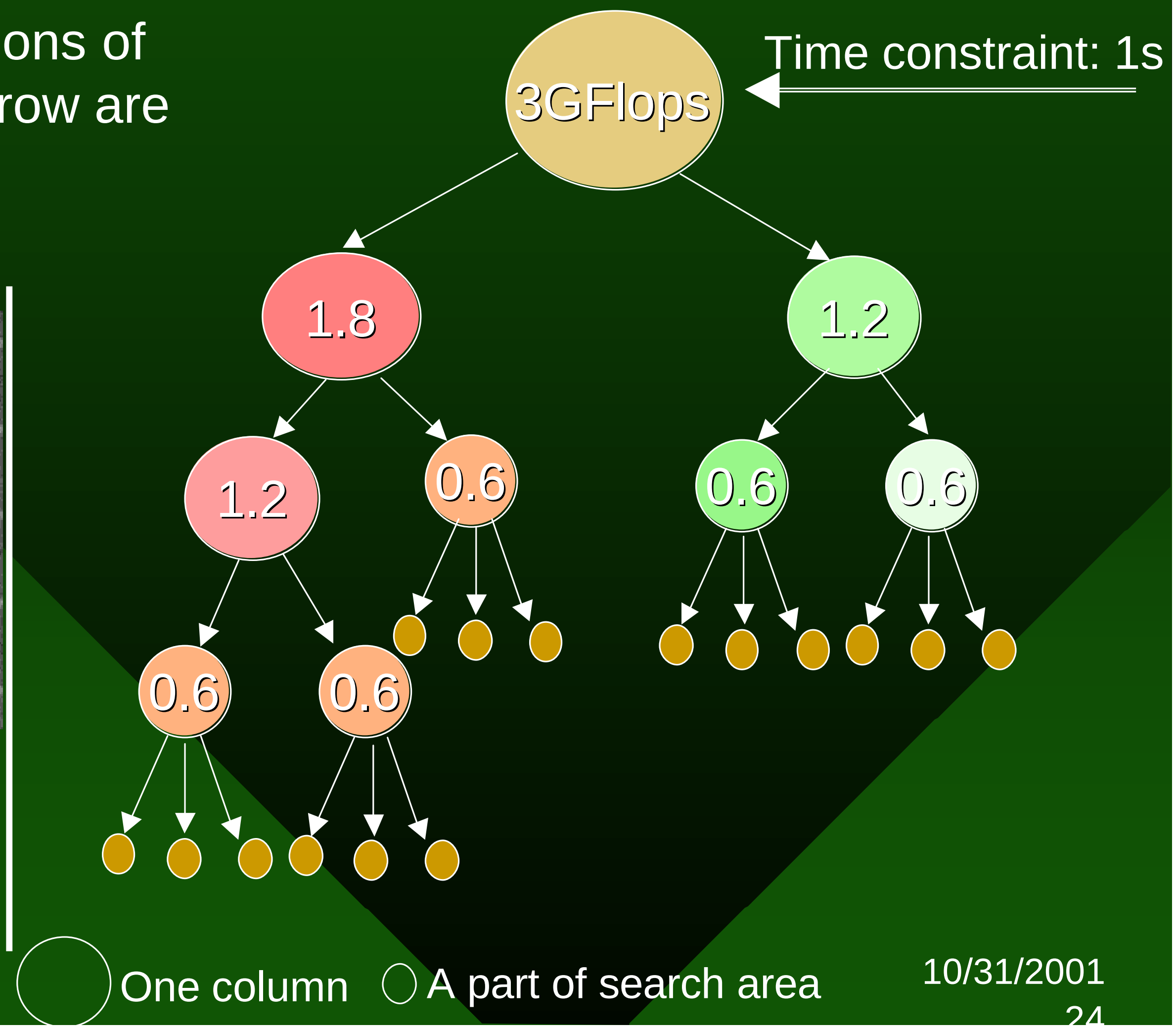
- Limit pattern search area on the tissue image
- Computing power needed depends on the size of pattern, can be reduced to few GFlop/s
- For the sample tissue: 3 GFlop/s needed

=> Should be parallelized.

The PDDS: Parallelization

•Assumption: The positions of patterns in the previous row are known





Conclusion

- We have shown a dynamic parallel object model:
  - Suitable for IHPC applications
  - Dynamic parallelism based on user requirement driven
  - Object oriented approach for developing IHPC applications
  - A case study using the model is presented
- A metacomputing object architecture to support the model