

# Passive Data Access for Data Intensive Computing

TUAN ANH NGUYEN, PIERRE KUONEN

University of Applied Sciences, Valais

## Agenda/Topics to Be Covered

- Data intensive computing issues
- Our interest
- Paroc++: a new programming paradigm
- Passive data access and the support for data intensive computing
- Example
- Summary

5/16/02

2

## Data intensive computing issues

- Large volume of data (petabytes scale)
- Distributed data computing
- Network bandwidth utilization
- Transparent and secure access
- Interaction with other databases and documents

5/16/02

3

## Our interest

We focus on:

- Large volume of data
- Distributed data computing
- High network bandwidth utilization

Applications:

- Parallel HPC application with data intensive communication
- Distributed processing environment
- Object oriented approach

5/16/02

4

## Our approach

### PAROC++: Parallel Object C++

- New concept of parallel object
- Super set of C++
- Some extension keywords: sync, async, conc, mutex, etc.
- Supporting library and infrastructure

### Our objectives:

- New programming paradigm based on parallel objects
- An object-oriented programming language
- Supporting parallel high performance computing

5/16/02

5

## Parallel object concept

- Object parallelism
- Parallel objects:
  - ◆ Can be located on different resources
  - ◆ Some operations on the same 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
- No distinction between object interface and object implementation
- Users access a parallel object as if it is a “local” object

5/16/02

6

## Parallel Object and Object Description

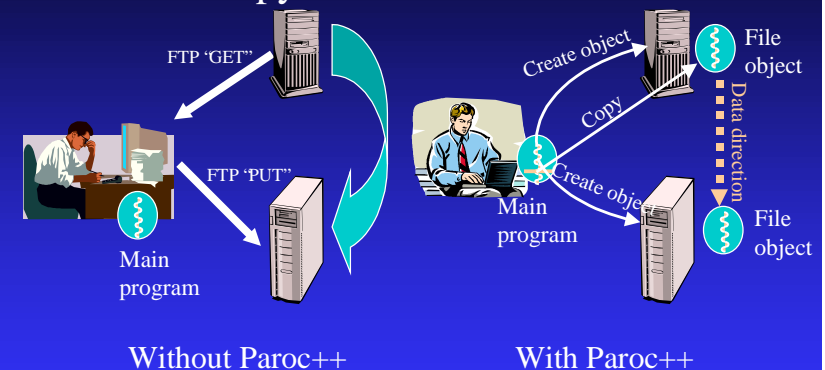
- Self-describable object: each parallel object has a user-specified object description
- Describing the requirement of parallel objects
- Will be used as a guideline for allocating resource
- Can be expressed in terms of:
  - ◆ Resource location (low-level)
  - ◆ Maximum computational power (e.g. Mflops)
  - ◆ Communication with other parallel objects
  - ◆ Memory needed

5/16/02

7

## Paroc++: An example

### • Remote file copy



5/16/02

8

# Paroc++: An example

```
parclass File {
public:
  File(char *hostname) @ { host=hostname; };
  int Open(char fname[256], int mode);
  int Read(out, size=n char *buf, int n);
  int Write(in, size=n char *buf, int n);
  int CopyTo(File &dest);
protected:
  int fd;
};
...
void main(int argc, char **argv)
{
  ...
  File f1("a303-sun01.hevs.ch");
  File f2("t0-p5.hevs.ch");
  ...
  f1.CopyTo(f2);
}
5/16/02
```

```
int File::Open(char *fname,
int mode
{ fd=open(fname,mode);
return (fd>=0);
}
int File::Read(char *buf,int n)
{ return read(fd,buf,n);
}
...

```

Object description

Optional parameter specification

# Paroc++ and data intensive computing

- Data processing elements will be presented by parallel objects
- Fully distributed approach, no centralized data management needed
- Network resource-aware system
- Provide two modes for data access: active mode and passive mode

# Traditional method

## Active data access

- User send "data acquisition" request to remote data source
- Remote data source process the request, acquire or compute the output data and send it to the user
- User wait for the returned data and then continue

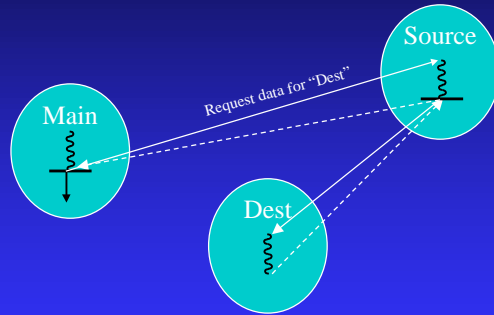
# New approach

## Passive data access

- User send the request and the target parallel object to the remote data source and then continue
- The remote data source will do some computation, generate the output data and put them directly to the target parallel object
- This data access mode is native support in Paroc++ by the ability of passing a parallel object as an argument of method invocation and of different invocation semantics

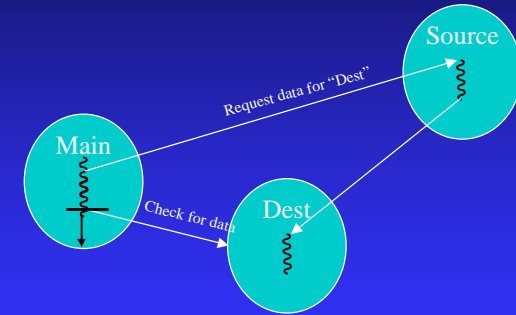
# Paroc++ data access methods

## ■ Synchronous/Sequential access



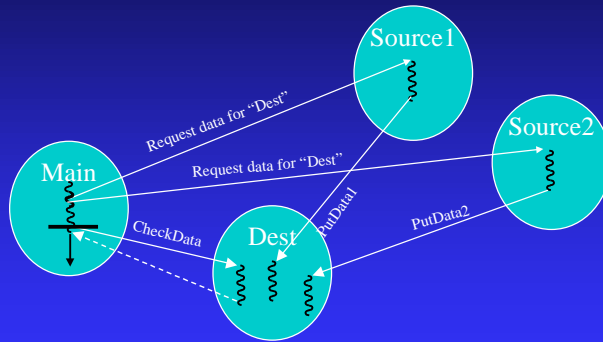
# Paroc++ data access methods

## ■ Asynchronous/Sequential access



# Paroc++ data access methods

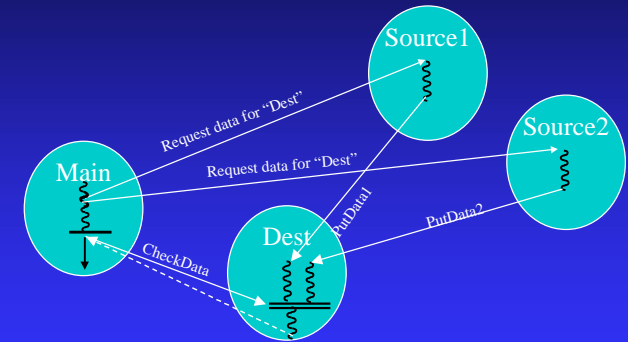
## ■ Asynchronous/Concurrent access



**sync conc** void CheckData(...);

# Paroc++ data access methods

## ■ Asynchronous/Concurrent access



**sync conc** void CheckData(...) **mutex** PutData1, PutData2;

# Passive data access for data intensive computing

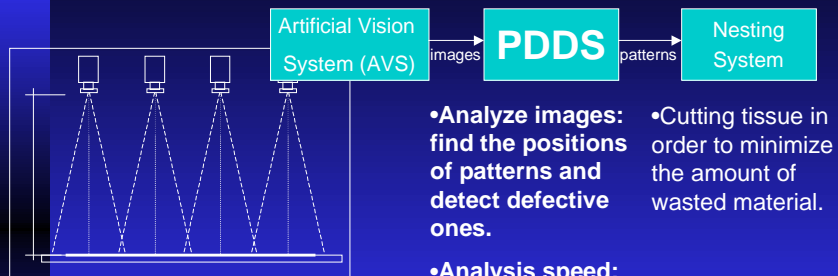
- Users can predict the data they need and acquire them prior to the data are really used
- Better overlapping between computation and communication
- One piece of data can be collected and synthesis automatically from multiple data sources
- Optimize the bandwidth usage

# Case study

## Pattern and Defect Detection System (PDDS)

- A part of ForAll project, an European project financed by Swiss Government
  - ◆ CTI project No 5130.1 in the EUREKA European program
  - ◆ Done in the collaboration with EPFL
- Find pattern positions and detect defects on tissue images, all in real-time
- Type of tissues: non-uniform tissues with rectangular patterns

# ForAll project (CTI project)

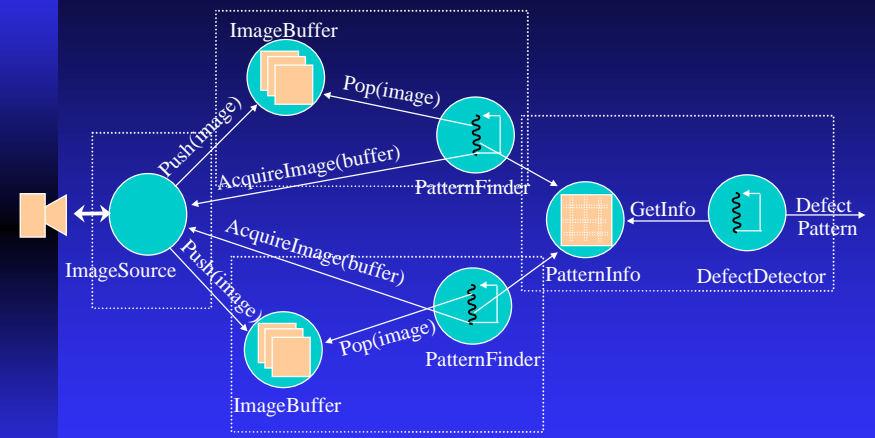


- Analyze images: find the positions of patterns and detect defective ones.
- Cutting tissue in order to minimize the amount of wasted material.
- Analysis speed: >3.3Mpixel/s.

### ■ Technical info

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

# From Paroc++ to ForAll



Parallel objects interaction

# Summary

## We have presented:

- Some issues in data intensive computing
- An object-oriented programming paradigm for parallel HPC applications
- A programming language Paroc++
- Passive data access in the view of Paroc++
- An example of pattern and defect detection (ForAll European project)

We conclude: The combination of Paroc++ and passive access can achieve the performance for data intensive computing