Summary

Task parallelism to exploit network parallel computing environments

Dynamicity leads to a run-time oriented approach

High level inter-task communication interface

Compile-time extensible run-time system

Modular run-time system design for environment customization

Other Work

Schooner (Homer, Schlichting, 1994)

Converse (Kale, et. al., 1995)

Opus (Haines, et. al., 1995)

(Alvani, et. al., 1995)

Current Status

Fx compiler as network task parallel compiler

Have run heterogenous task parallel Fx programs

Preliminary run-time system implementation

Distribution Subsystem

Computes address relations

Caching strategy

Binding Subsystem

Task mappings

Data distributions

Address computation functions

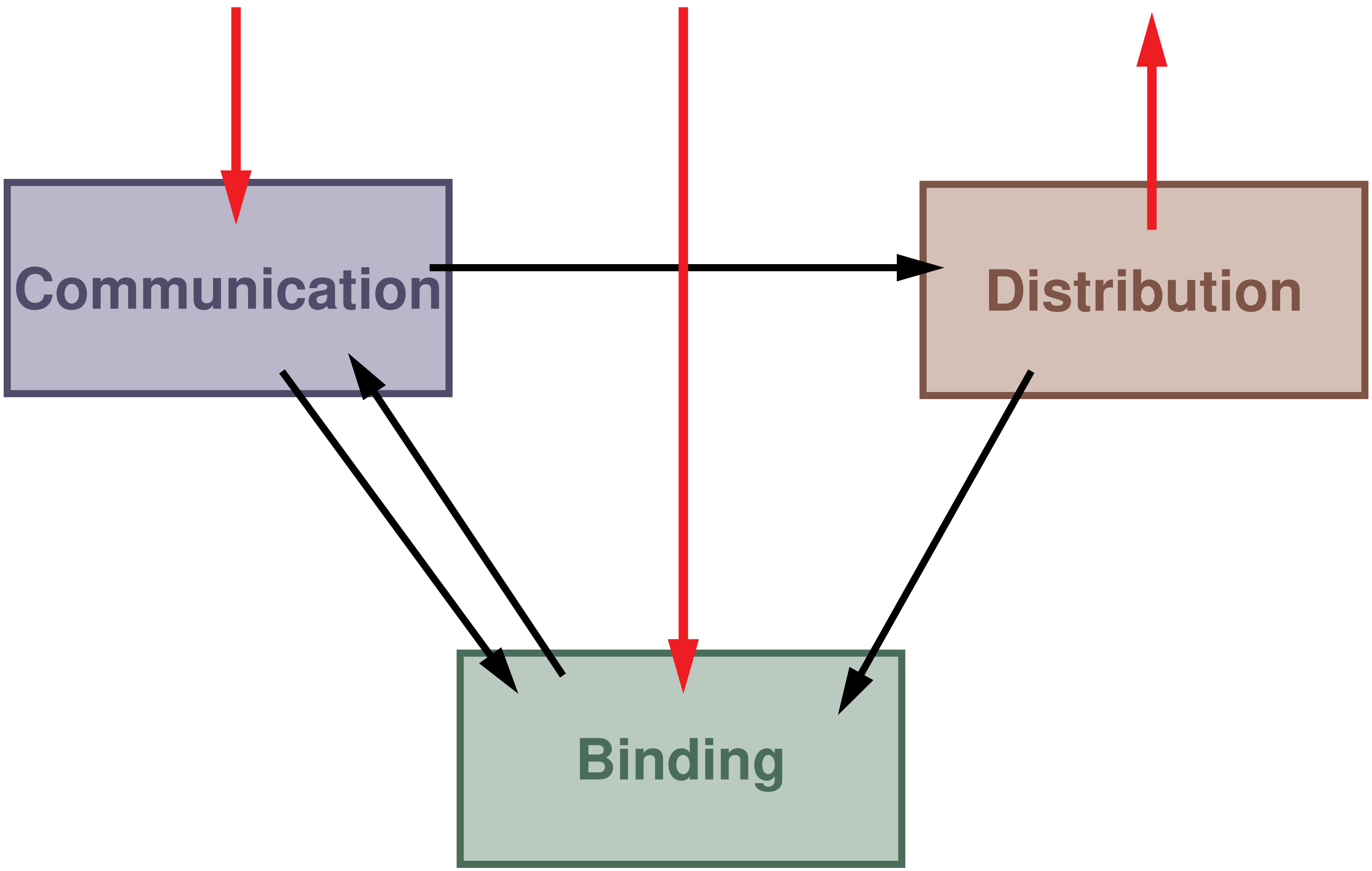
Communication Subsystem

Inter-task communication interface

Scheduling

Parallel data transfer

Run-time System Modularity



Custom Address Computation Functions

Generated by compilers/programmers

Registered by network task parallel compiler

Run-time system will call them instead of library functions

Exploit compile-time knowledge

Easy extension of supported data distribution types

An Inter-task Send

Compiler

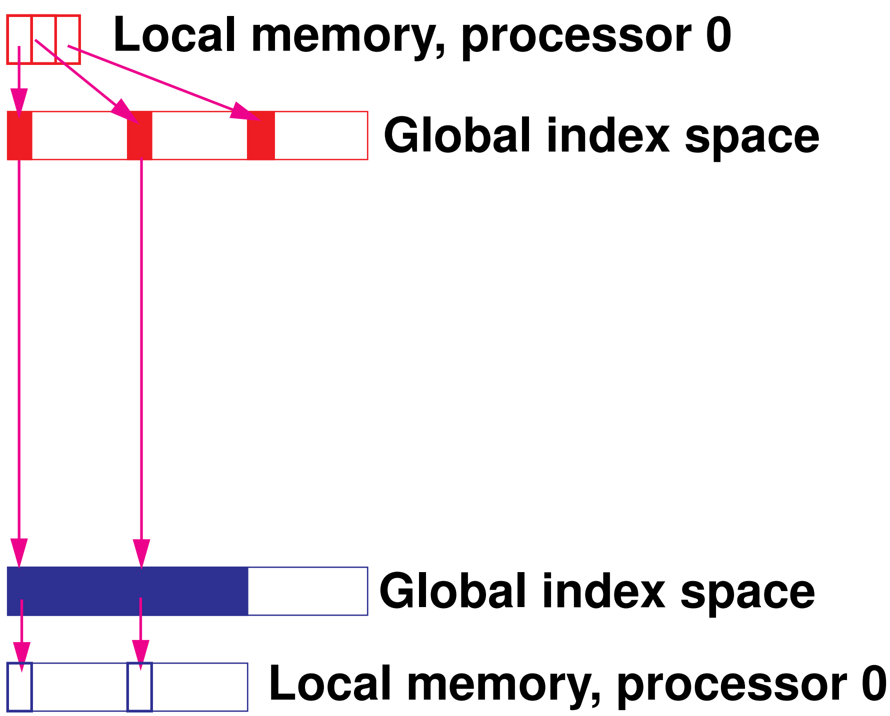
Register endpoint tasks
Register endpoint distribution types
Register any custom address computation functions
Call inter-task send
Endpoint task tags
Distribution type tags

Run-time system

Bind endpoint task tags to mappings
Bind distribution type tags to current distributions
Bind endpoint distribution type tags to an address computation function
Compute address relations
Schedule data transfer
Perform data transfer

Address Computation

HPF block-cyclic vector to DOME load-balanced block vector



Address relation **between each pair of source and destination processors**

Address computation **is process of computing these relations**

An *address computation function* computes a relation given distributions and processor numbers

High-level Run-time System Interface

Inter-task communication interface

Details hidden from interface user

- Task mapping
- Data distributions
- Processor and network heterogeneity

Registry interface

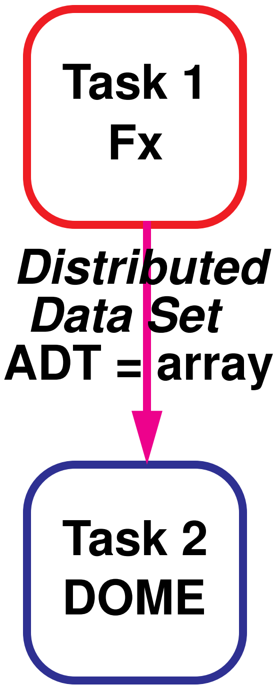
- Initial task mappings, data distribution types, etc.
- Custom address computation functions

Custom address computation function interface

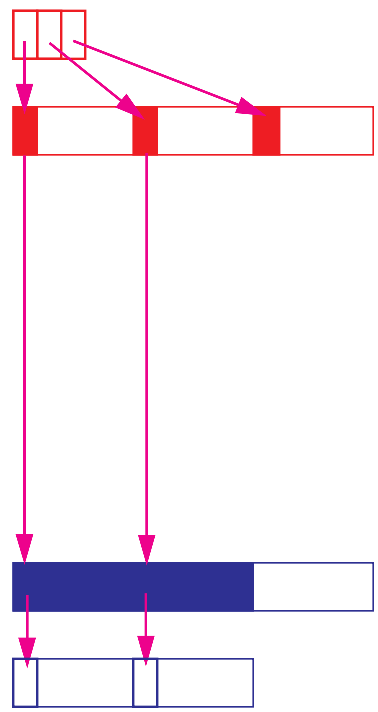
- Standard calling convention for compiler supplied functions

Communication Example

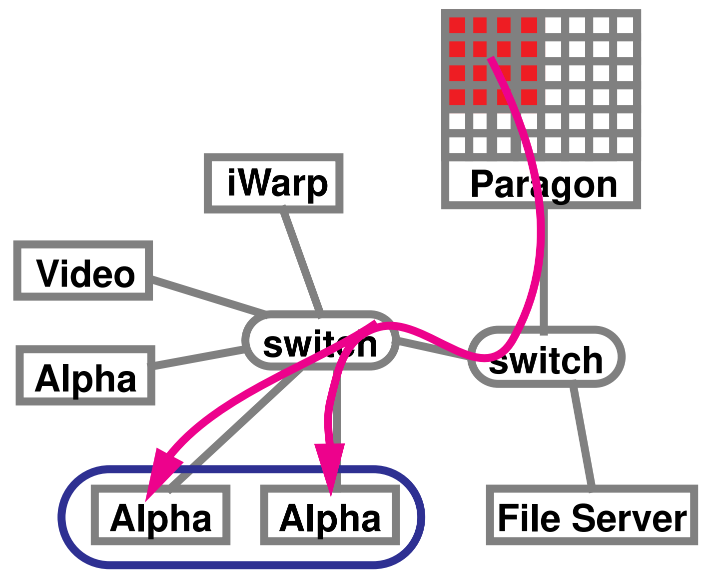
Compiler View



Address Computation



Data Transfer



Network Task Parallel Compiler

Generates run-time system initialization calls

Maps tasks to the environment (inital mapping)

Mapping can change (global load balancing)

Analyzes inter-task dependencies

Inter-task dependencies never change

Generates RTS calls for communication between tasks

Inter-task send of ADT-compatible distributed data sets abstraction

Compilation Inputs

Parallel procedures

Appropriate compilers

Task interface information

Environment configuration

Computation and communication resources

Tasking program

Task interaction
Mappings

Custom address computation functions

Compiler or programmer generated

Task Parallel Model

Tasks are invocations of parallel procedures

Written in any language

Tasks send and receive distributed data sets

Abstract data type of sending and receiving task must match
Representations can differ

Example

HPF Parallel Procedure

```
subroutine foo(a,b)
real a(1024,1024), b(1024,1024)
distribute a(*,BLOCK), b(BLOCK,*)
input  a
output b
...
end
```

Task

```
receive(ARRAY,HPF,...,A)
foo(A,B)
send(ARRAY,HPF,...,B)
```

Our Approach

Task parallel model

Compilation inputs

Network task parallel compiler

Run-time system

- High level run-time system interface
- Compile-time extensibility
- Modularity

Meeting the Challenges

Multiple languages and paradigms

Use the language that is best for the machine and algorithm

Dynamic and/or easily configurable program mappings

Challenges

Architecture and network heterogeneity

Expressability

Will a single language suffice?

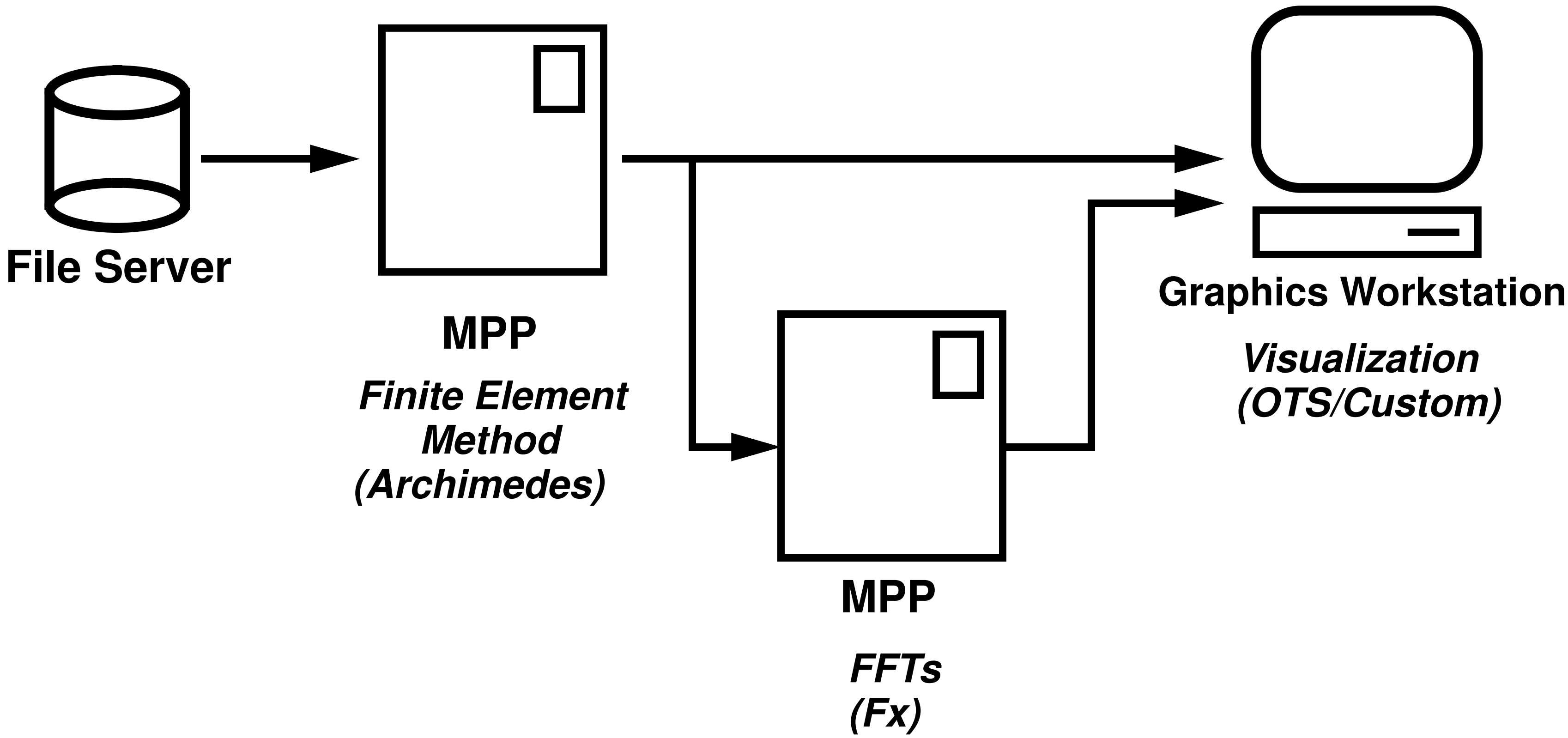
Dynamic environment

Other users
Other network traffic

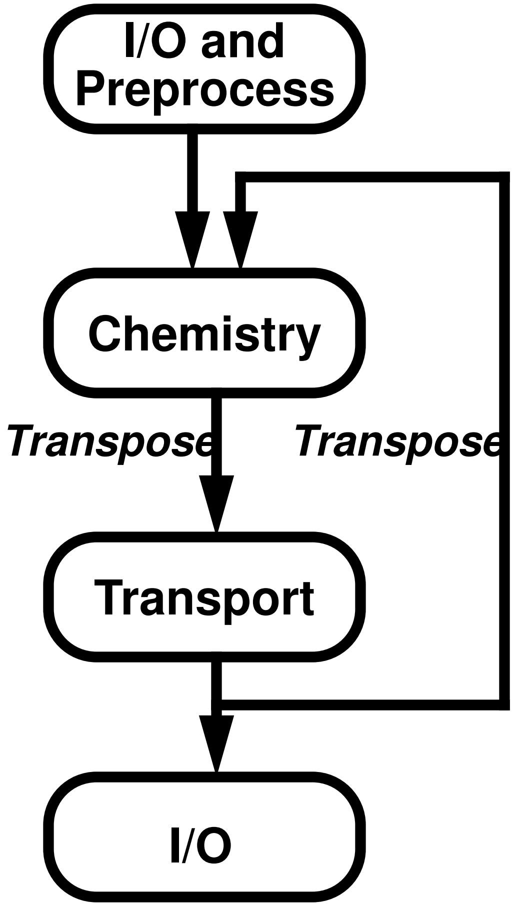
Performance portability

Network environments differ far more than MPPs

Earthquake Ground Motion Modeling



Air Quality Modeling

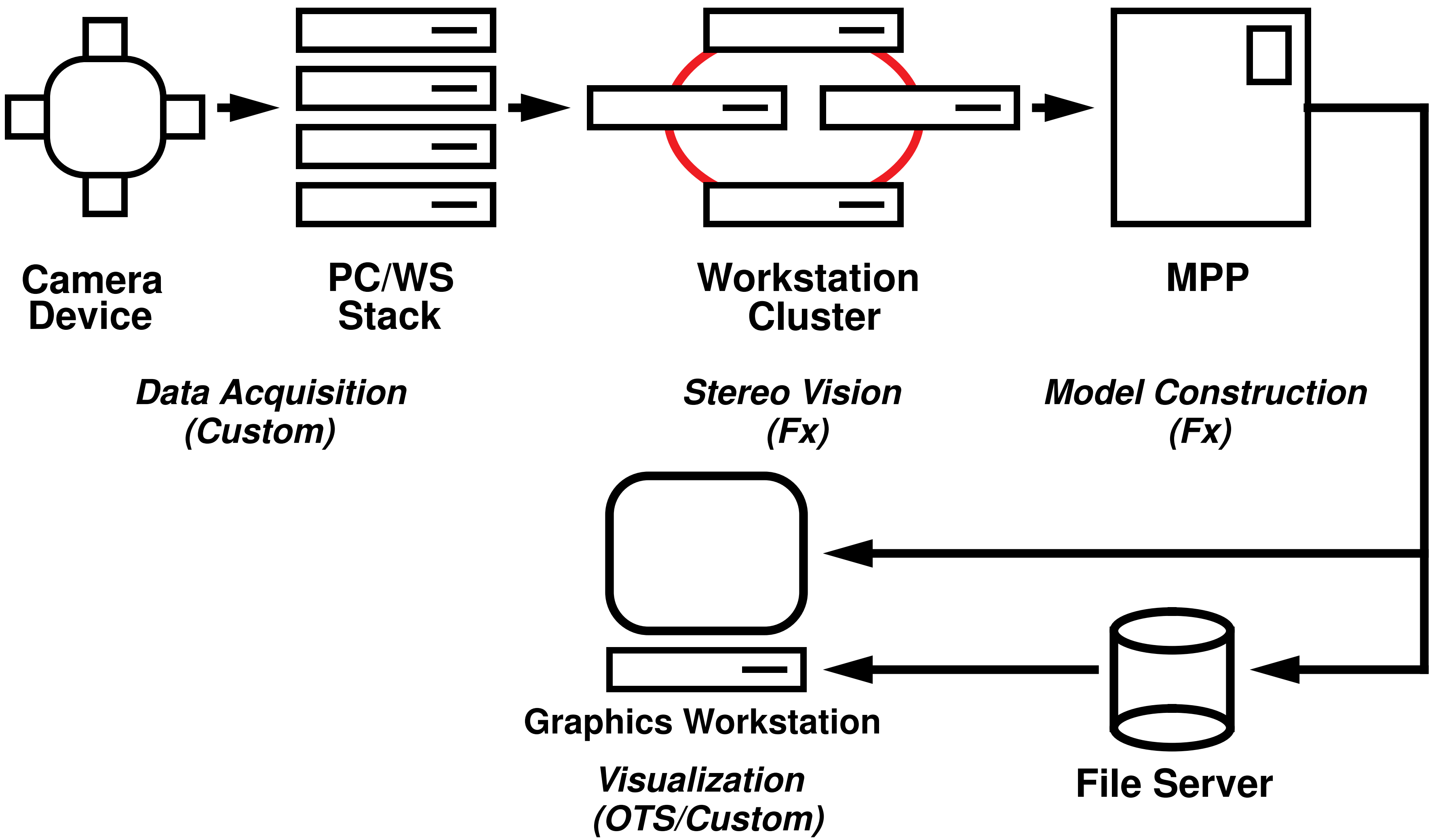


Mapping is dependent on application and environment parameters

Want to exploit distributed environment to allow largest possible models

Performance portability

Real-world Modeling



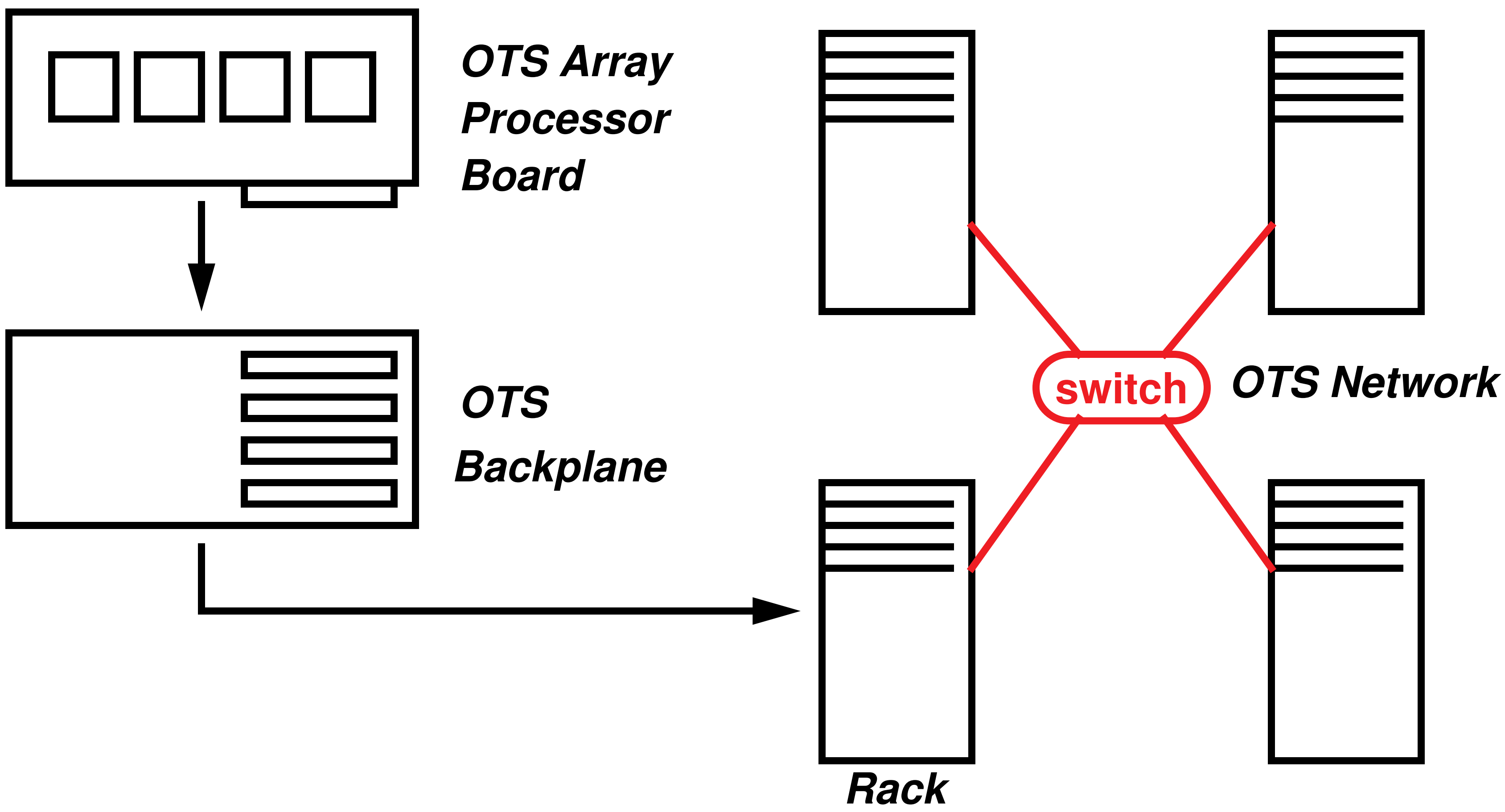
Applications

Real-world modeling

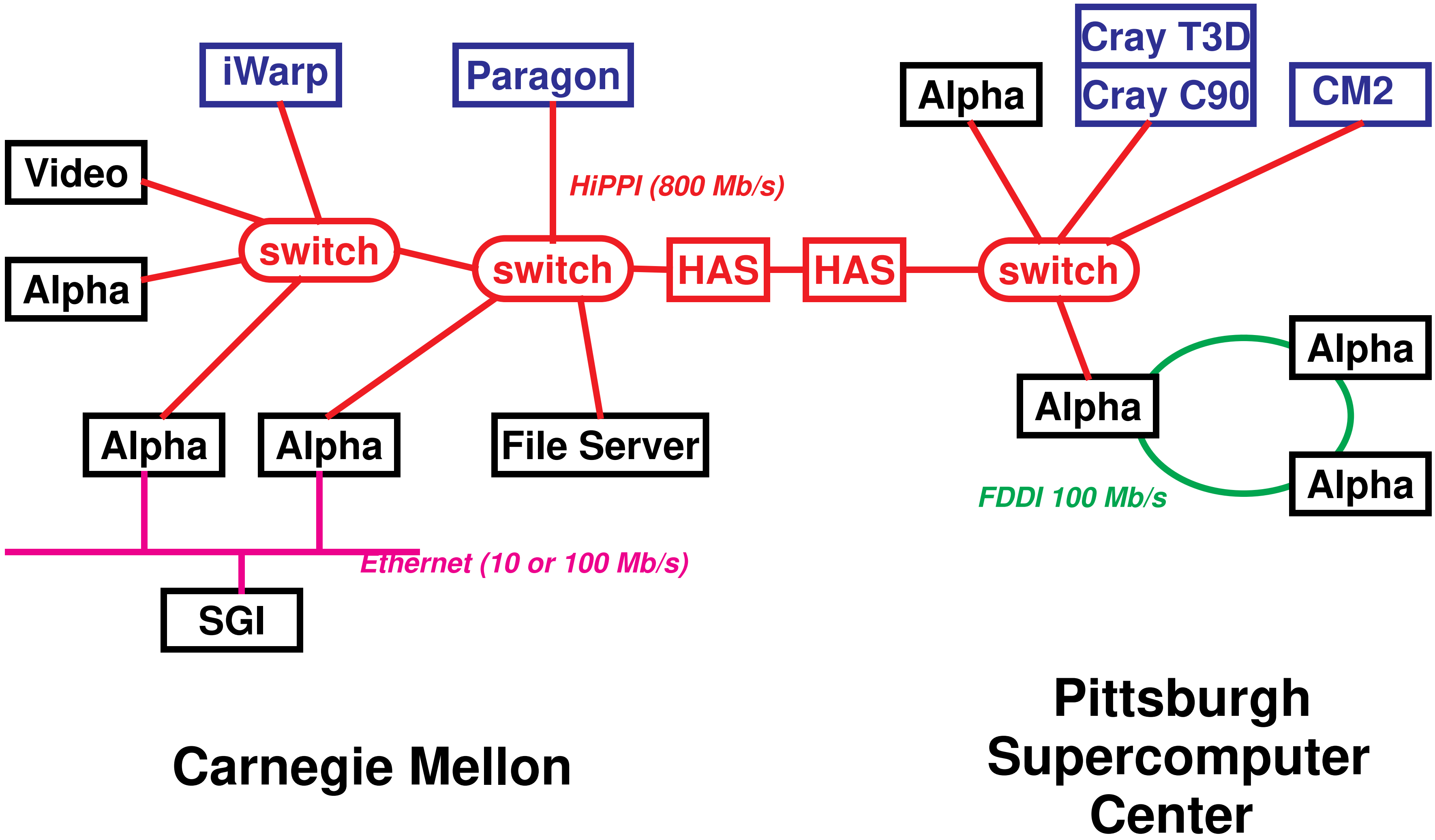
Air quality modeling

Earthquake ground motion modeling

Off-the-shelf Systems



CMU's Network Parallel Computing Environment



Motivations for Network Parallel Computing

Hardware

Diverse computing resources
Fast general purpose networks

Applications

Resource requirements
Phases

Outline

Motivations

Challenges

Approach

Language and Run-time Support for Network Parallel Computing

Peter A. Dinda David R. O'Hallaron Jaspal Subhlok
Jon A. Webb Bwolen Yang
School of Computer Science
Carnegie Mellon University

http://www.cs.cmu.edu/~fx