Mobile Code for Component Customization and Optimization

CoreGRID WP3 Meeting, June 2005, Barcelona, Spain

University of Münster, Germany
Jan Dünnweber

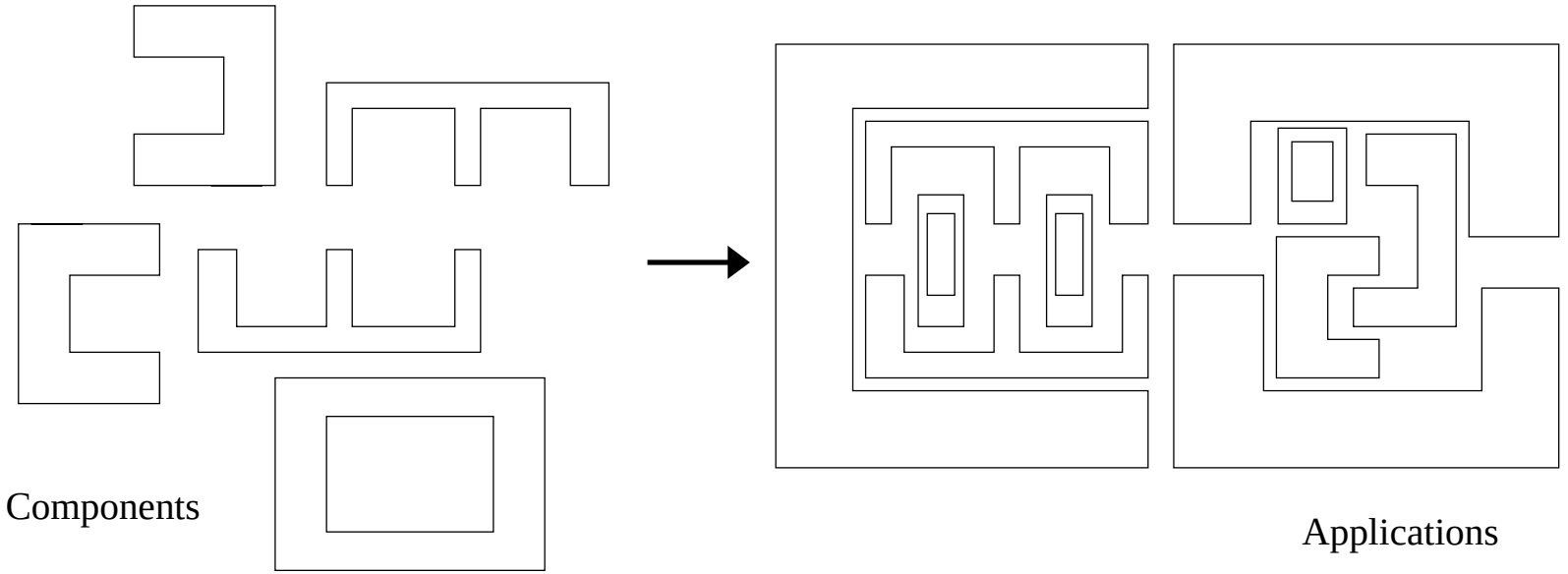
Software Components for Grid Programming

Design:

- Repository provides a few, simple, ready-made components
- Complex structures are build by nesting and combination

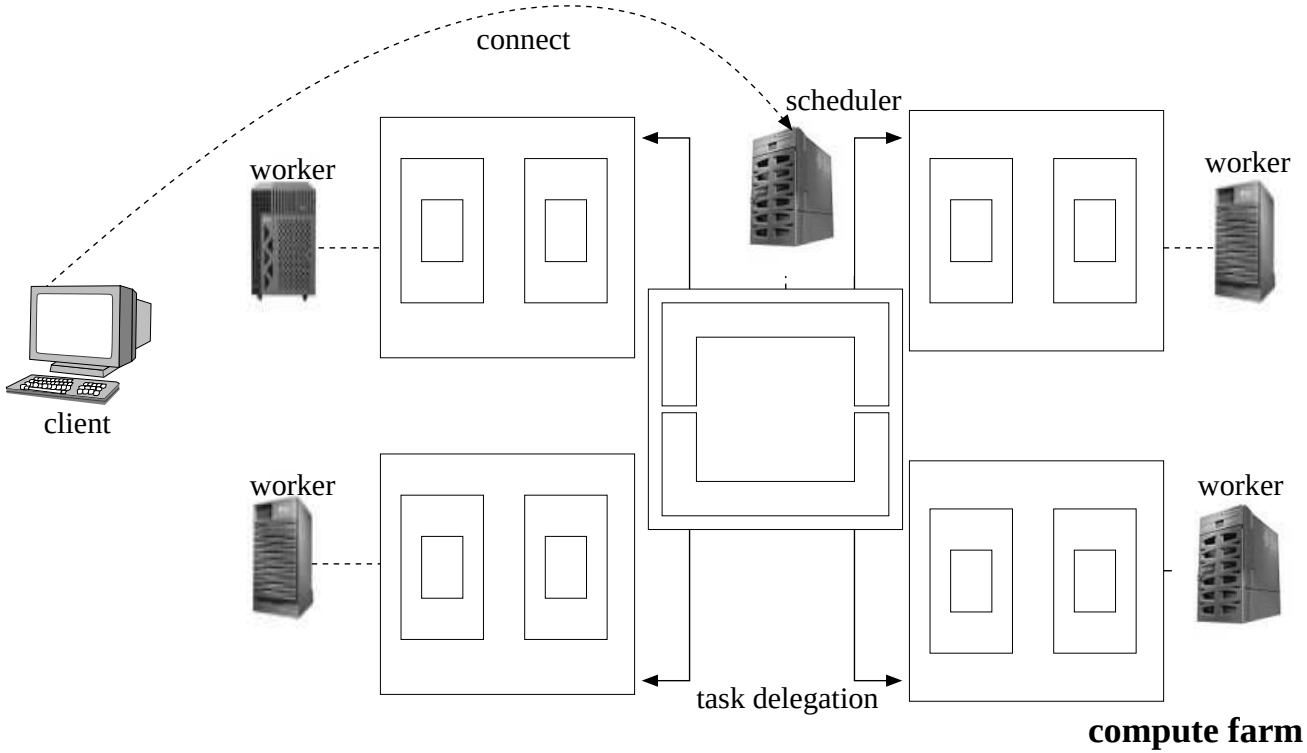
Vision:

- Grid programming becomes as easy as stacking building blocks



Topologies for recurring patterns can be prearranged

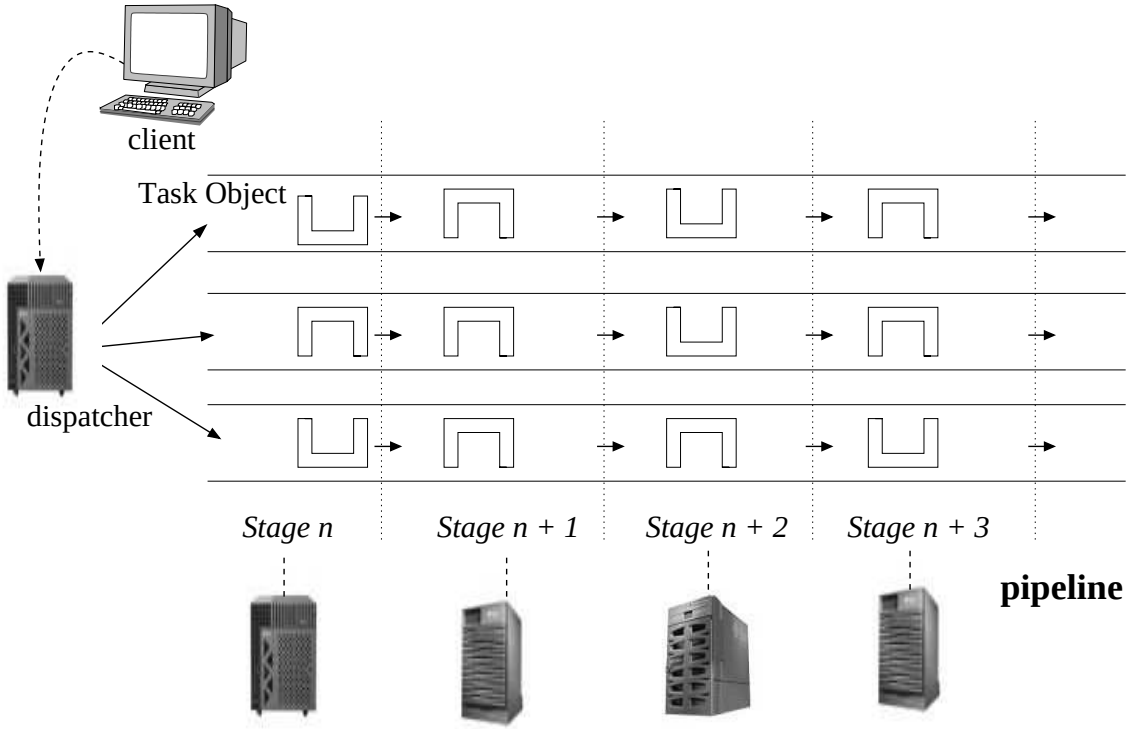
- Example 1: The *compute farm*
  - ➜ One server hosting the scheduler component
  - ➜ Various additional servers hosting worker components



COMPONENT DISTRIBUTION OVER THE GRID

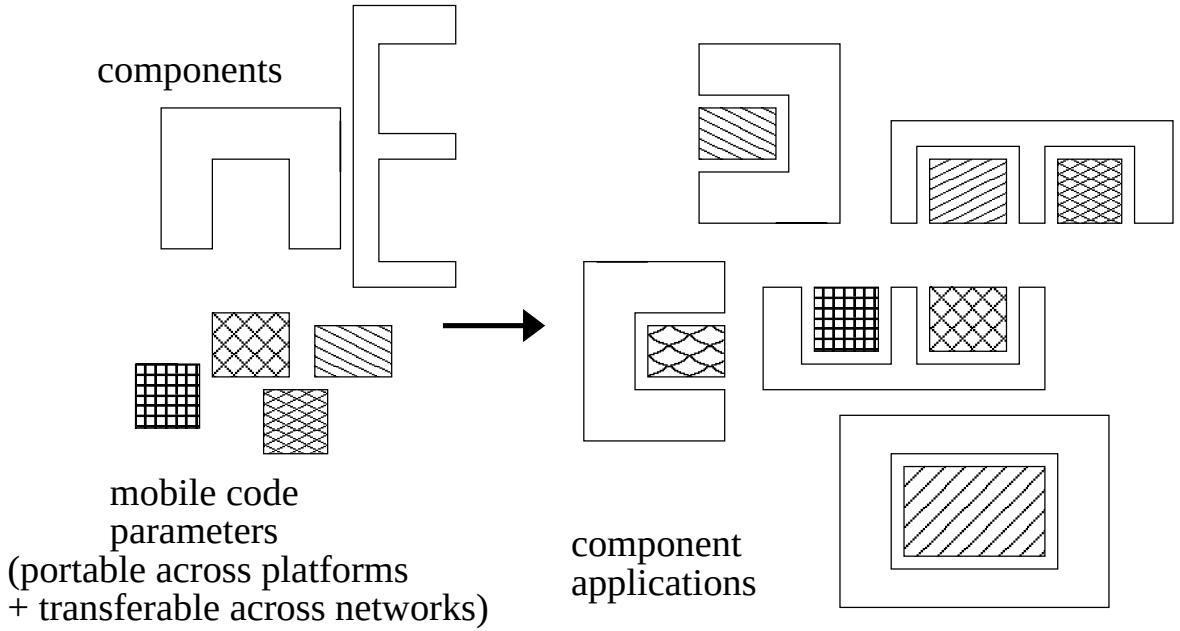
Component applications are optimally distributed

- Example 2: The *pipeline*
  - ➜ One server dispatches incoming tasks
  - ➜ Various additional servers process data in successive stages



Mobile Code for Component Customization

- Reusable components are implemented *partially* and expect application specific code, provided via parameters
  - ➜ *mobile code* is required,
    i. e. code that is *transferable* and *portable*



Example Application 1: Bioinformatics

- Protein sequence distances are described using *similarity matrices* with elements defined as follows:

$$s_{i,j} := max\left(s_{i,j-1} + penalty, s_{i-1,j-1} + \delta(i,j), s_{i-1,j} + penalty\right)$$

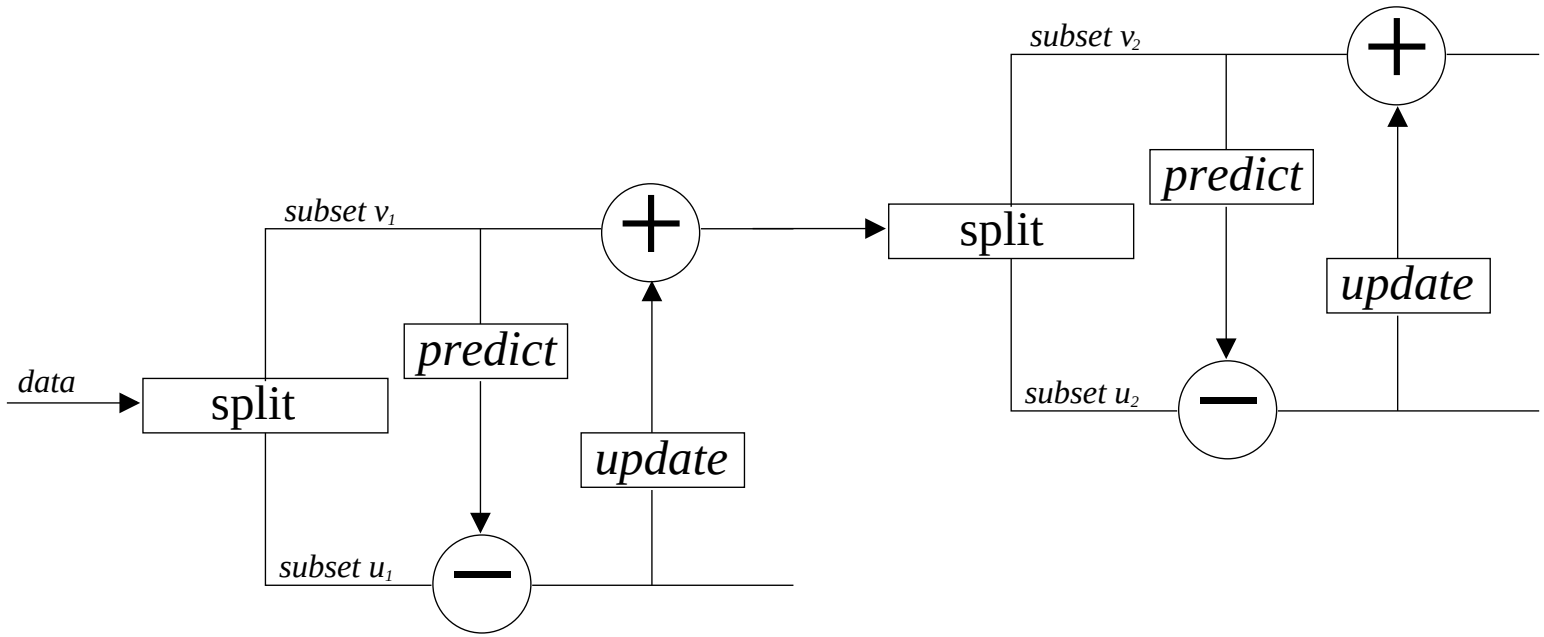
wherein

$$\delta(i,j) := \begin{cases} +1 & \text{, if } \epsilon_1(i) = \epsilon_2(j) \\ -1 & \text{, otherwise} \end{cases}$$

- Computations of this kind are called *global alignment*
- Similarity definition is expressed via mobile code parameters
- Matrix is calculated using a *compute farm* component

EXAMPLE APPLICATION 2: WAVELET TRANSFORM

- *DWT* is a recurring, compute intensive operation in image processing, data compression etc.
- The *lifting*-algorithm defines the transform via repeated application of three functions called *split*, *predict* and *update*



➜ Transform is programmed using a pipeline component

➜ *split*, *predict* and *update* are expressed via mobile code

Problem 1: Fixed Component Behavior

➜ In the bioinformatics example, every element depends on north-, northwest- and west-neighbor

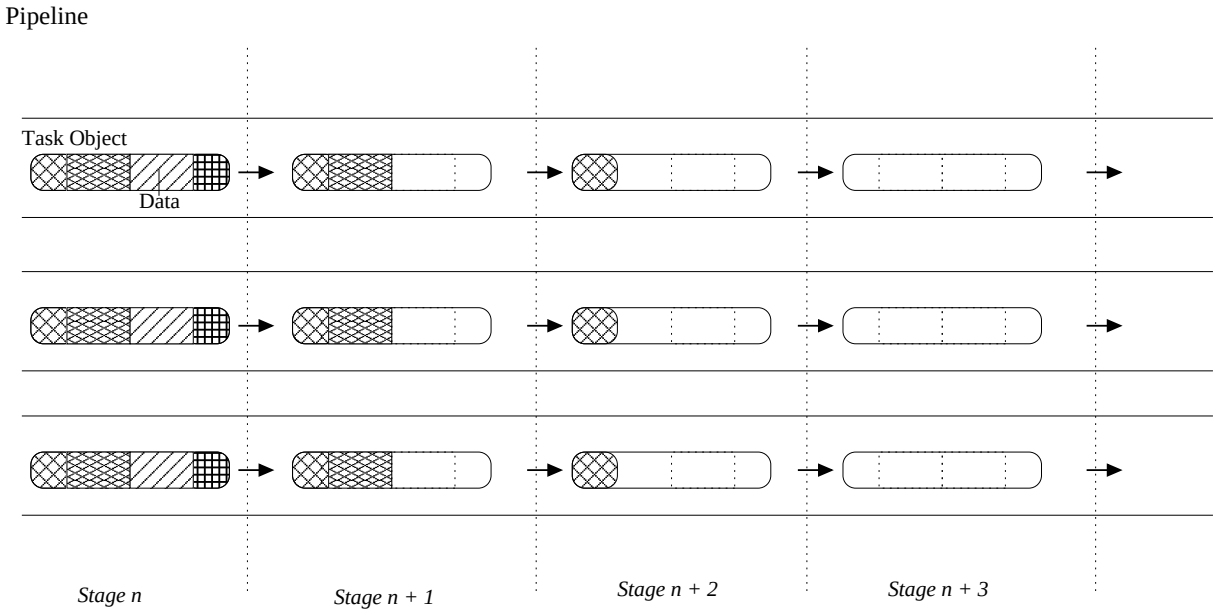
		G	G	A	C	T
	0	−2	−4	−6	−8	−10
G	−2	1	−1	−3	−5	−7
T	−4	−1	0	−2	−4	−4
T	−6	−3	−2	−1	−3	−3
C	−8	−5	−4	−3	0	−2
T	−10	−7	−6	5	−2	1

➜ Independent Elements are positioned along the matrix' antidiagonals

➜ A *compute farm* will calculate the matrix as one atomic task anyway

Problem 2: Suboptimal Component Arrangements

- In the wavelet application example, the input data is bisected in each pipeline stage



- Disadvantage: finished tasks will unnecessarily be passed through numerous remaining pipeline stages
- Inefficiency: reduced degree of parallelism
- Problem: tasks with a varying number of stages do not fit into the pipeline model

Solution: Mobile Code for component optimization

- The bioinformatics example can be optimized by prepartitioning the input data according to the *wavefront pattern*
- The wavelet application can be optimized by *skipping stages* in the pipeline, once a task is finished
- Both optimizations require a pre- or post-processing of data that is independent from the standard farm worker resp. pipeline stage operations
- Such optimizations can be expressed using mobile code parameters that are applied by the components before resp. after each single operation

Features of our Component Models

feature	1	2	3	4	5	6	7	8
ProActive	yes		ongoing	yes	ongoing	yes	yes	yes
Reflex		yes		yes				
QUB	yes	yes	yes		yes			
HOC-SA	yes	yes	yes		yes		yes	yes
Polytope	yes	yes	yes		yes	yes		yes
ASSIST	yes	yes	yes		yes	yes		
MALLBA			yes		yes	yes		
GAT	yes	yes	yes				yes	yes
Ibis	yes		yes			yes		via GAT

A WSRF compliant service definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="MasterService"
                  targetNamespace="http://org.gridhocs/Master"
      xmlns:wsdlpp="http://www.globus.org/namespaces/2004/10/WSDLPreprocessor"
                  ...   <!-- more namespace declarations          -->
                  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:import location="../../wsrf/properties/WS-ResourceProperties.wsdl"/>
                   ...   <!-- more WSR-import statements          -->
  <wsdl:types>
    <schema targetNamespace="http://org.gridhocs/Master"
            xmlns="http://www.w3.org/2001/XML
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="ArrayOf_xsd_double">...
   </complexType>
      ...               <!-- more parameter type and              -->
      </element>        <!-- resource property declartions        -->
    </schema>
  </wsdl:types>

  <wsdl:message name="configureRequest">
    <wsdl:part name="in0" type="impl:ArrayOf_xsd_string"/>
  </wsdl:message>
                        <!-- more message declarations            -->

  <wsdl:portType name="MasterPortType"
                 wsdlpp:extends="wsrpw:GetResourceProperty
                 wsrlw:ImmediateResourceTermination"
                 wsrp:ResourceProperties="tns:MasterResourceProperties">
    <wsdl:operation name="configure" parameterOrder="in0">...
    </wsdl:operation>
    ...                 <!-- more operation declarations          -->
 </wsdl:portType>
</wsdl:definitions>
```

Parameter types are specified using XML-Schema

➜ No support for mobile code

Conclusion

- Using XML-Schema solely is insufficient for specifying flexible grid components and their interfaces
- External references are required to support mobile code
- Higher-Order Components (HOCs) are a workaround in the context of Java
- HOCs support Java Bytecode or scrips as a format for mobile code
- A programming language independent grid component model must support even more formats for expressing customizations and optimizations
- We propose to allow for arithmetic expressions, graphical specifications, etc.