ParCo 2005
Malaga, Spain

Building Interoperable Grid-aware ASSIST Applications via Web Services

M. Aldinucci
M. Danelutto, A. Paternesi,
R. Ravazzolo, M. Vanneschi

ISTI - CNR, Pisa, Italy
University of Pisa, Italy

ParCo 2005
Malaga, Spain

Building Interoperable Grid-aware ASSIST Applications via Web Services

M. Aldinucci
M. Danelutto, A. Paternesi,
R. Ravazzolo, M. Vanneschi

ISTI - CNR, Pisa, Italy
University of Pisa, Italy

ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"

IN SUPREMÆ DIGNITATIS 1343

CoreGRID

Grid.it project

Outline

- Motivating ...
  - high-level programming for the grid
  - application adaptivity for the grid
- ASSIST basics
- WS as transport in ASSIST apps
- Grid.it components & WS ports
- Concluding remarks

The grid

*"... coordinated resource sharing and problem solving in **dynamic**, multi institutional virtual organizations."*

Foster, Anatomy

"1) coordinates resources that are not subject to centralized control ..."
*"2) ... using **standard**, open, general-purpose protocols and interfaces"*
*"3) ... to deliver nontrivial **qualities of service**."*

Foster, What is the Grid?

The grid

*"... coordinated resource sharing and problem solving in **dynamic**, multi institutional virtual organizations."*

Foster, Anatomy

"1) coordinates resources that are not subject to centralized control ..."
*"2) ... using **standard**, open, general-purpose protocols and interfaces"*
*"3) ... to deliver nontrivial **qualities of service**."*

Foster, What is the Grid?

Moreover, since this is not **SeqCo**, I assume applications we are focusing on should be **parallel** (and hopefully **high-performance**).

// progr. & the grid

- concurrency exploitation, concurrent activities set up, mapping/scheduling, communication/synchronization handling and data allocation, ...
- manage resources heterogeneity and unreliability, networks latency and bandwidth unsteadiness, resources topology and availability changes, firewalls, private networks, reservation and jobs schedulers, ...

// progr. & the grid

- concurrency exploitation, concurrent activities set up, mapping/scheduling, communication/synchronization handling and data allocation, ...
- manage resources heterogeneity and unreliability, networks latency and bandwidth unsteadiness, resources topology and availability changes, firewalls, private networks, reservation and jobs schedulers, ...

... and a non trivial QoS for **applications**

// progr. & the grid

- concurrency exploitation, concurrent activities set up, mapping/scheduling, communication/synchronization handling and data allocation, ...
- manage resources heterogeneity and unreliability, networks latency and bandwidth unsteadiness, resources topology and availability changes, firewalls, private networks, reservation and jobs schedulers, ...

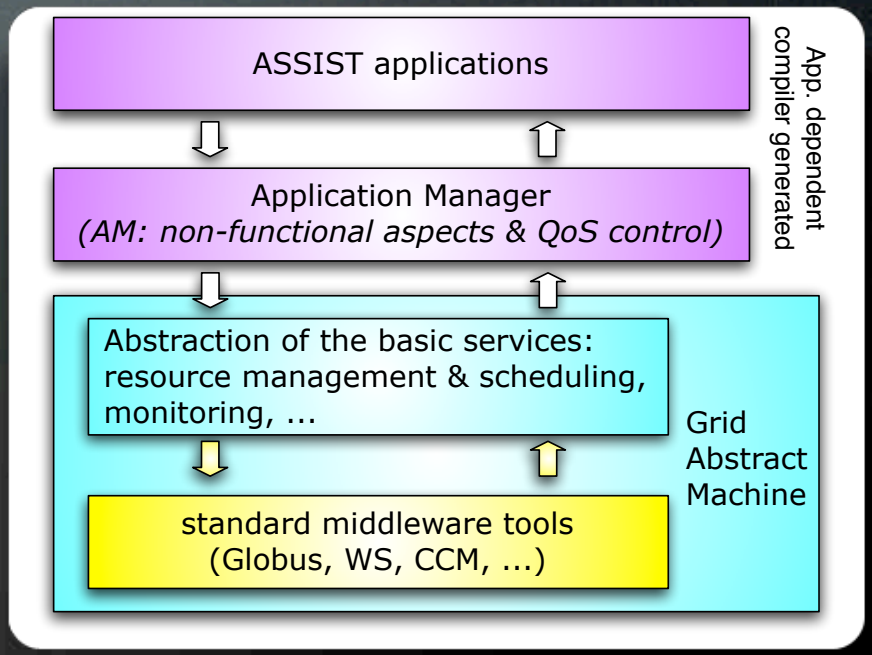
... and a non trivial QoS for **applications**

not easy leveraging only on middleware

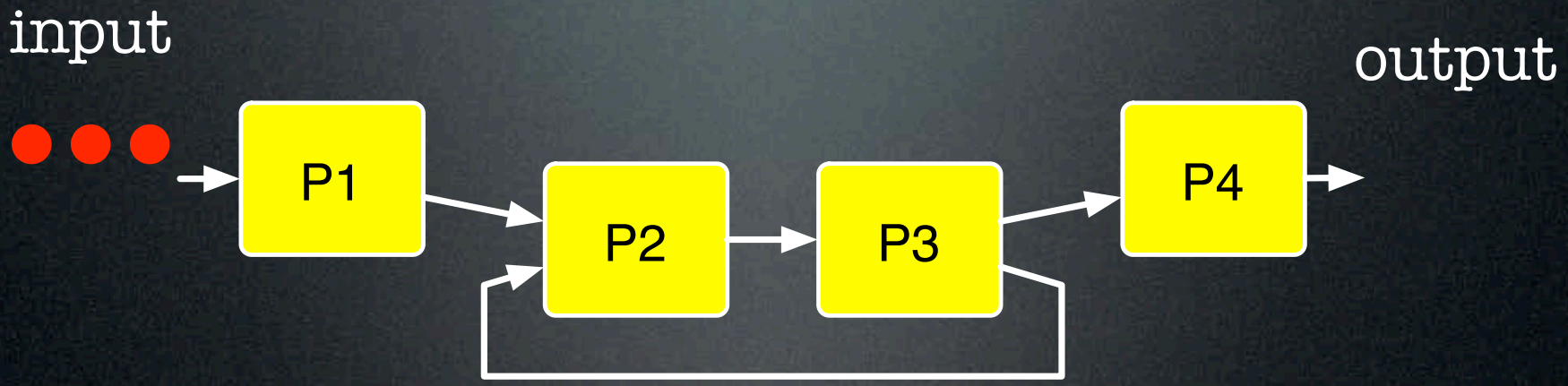
ASSIST idea

ASSIST is a high-level programming environment for grid-aware // applications. Developed at Uni. Pisa within several national/EU projects. First version in 2001. Open source under GPL.

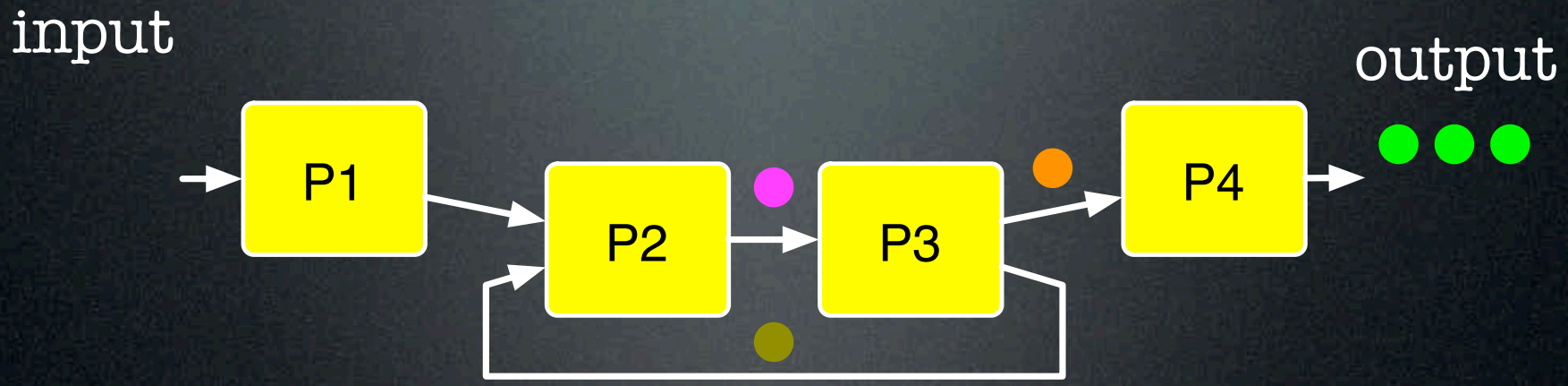
"moving most of the Grid specific efforts needed while developing high-performance Grid applications from programmers to grid tools and run-time systems"



app = graph of modules

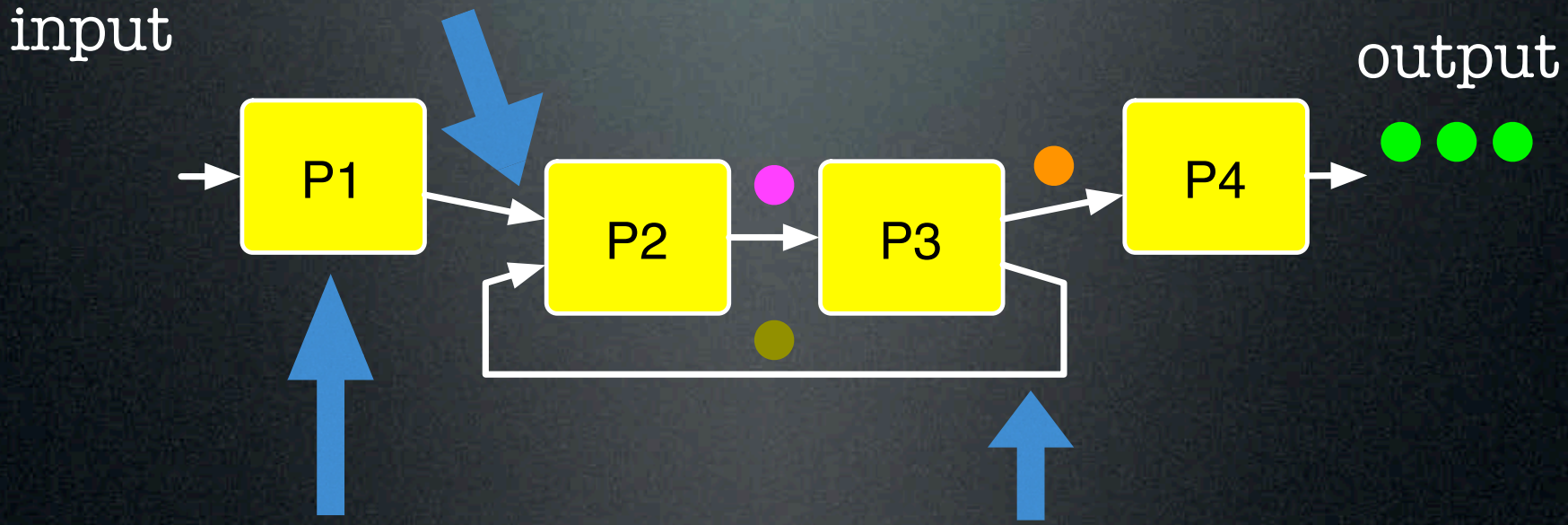


app = graph of modules

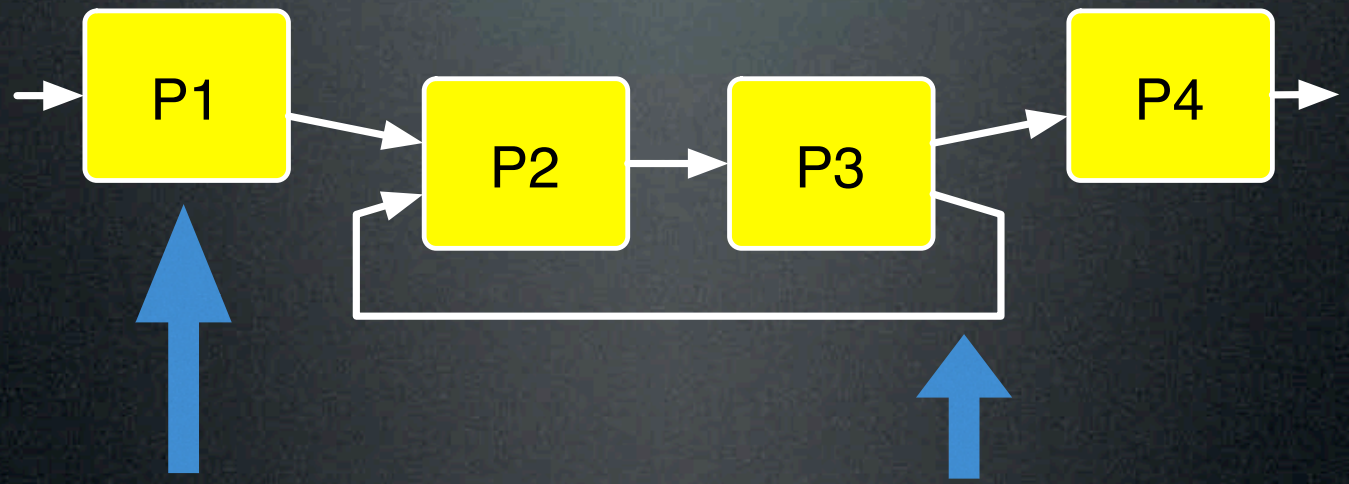


app = graph of modules

Programmable, possibly nondeterministic input behaviour



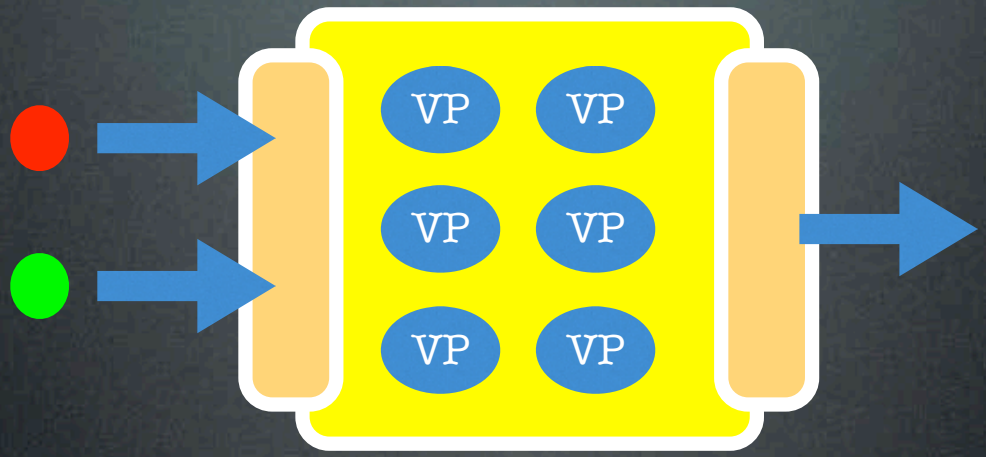
native + standard



ASSIST native or wrap
(MPI, CORBA, CCM, WS)

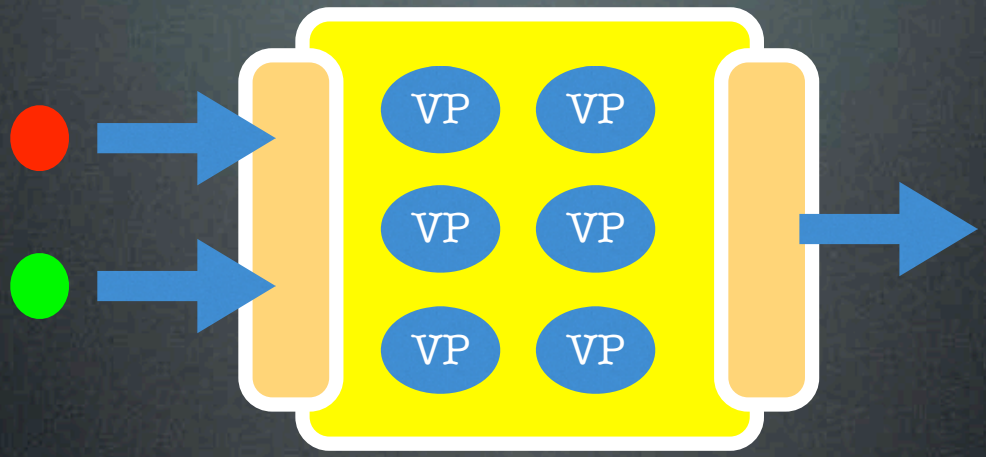
TCP/IP, Globus,
IIOP CORBA,
HTTP/SOAP

ASSIST native parmod



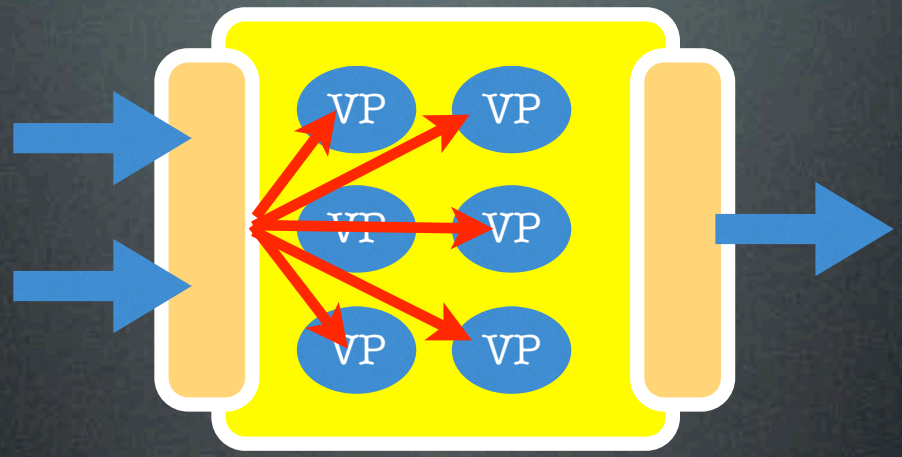
ASSIST native parmod

An “input section” can be programmed in a CSP-like way



ASSIST native parmod

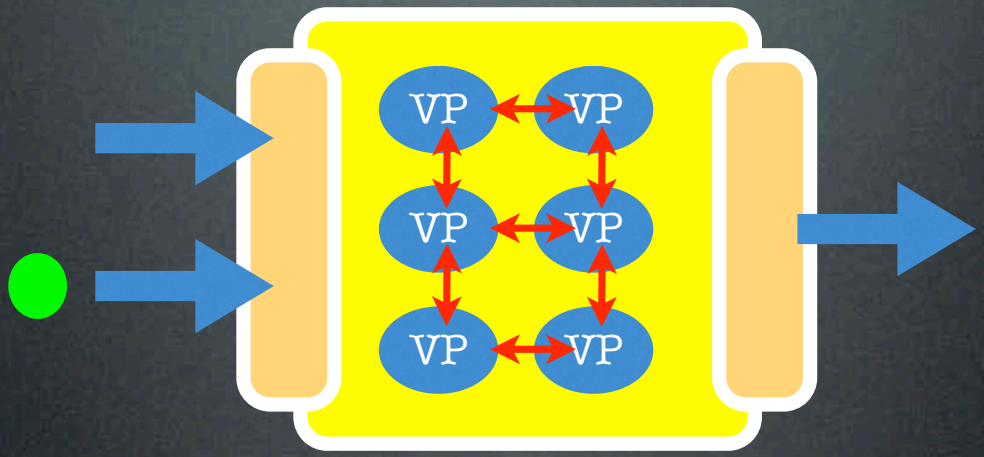
An “input section” can be programmed in a CSP-like way



Data items can be distributed (scattered, broadcasted, multicasted) to a set of **Virtual Processes** which are named accordingly to a topology

ASSIST native parmod

An “input section” can be programmed in a CSP-like way



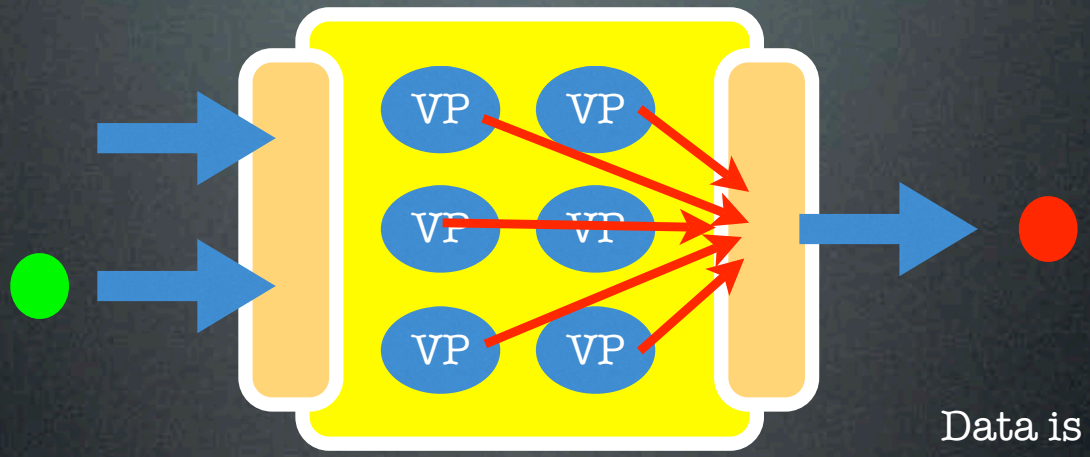
Data items can be distributed (scattered, broadcasted, multicasted) to a set of **Virtual Processes** which are named accordingly to a topology

Data items partitions are elaborated by VPs, possibly in iterative way

```
while(...)
  forall VP(in, out)
  barrier
```

data is logically shared by VPs (owner-computes)

ASSIST native parmod



An “input section” can be programmed in a CSP-like way

Data items can be distributed (scattered, broadcasted, multicasted) to a set of **Virtual Processes** which are named accordingly to a topology

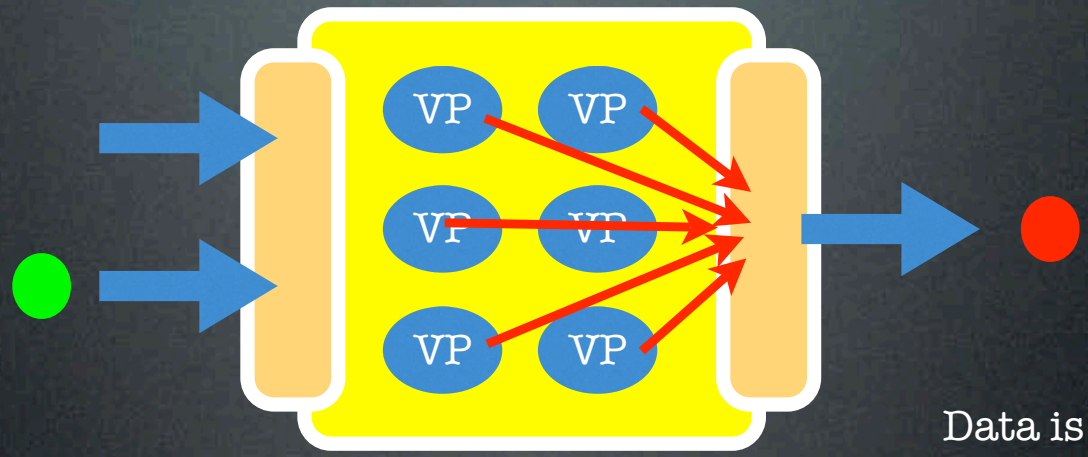
Data items partitions are elaborated by VPs, possibly in iterative way

```
while(...)
  forall VP(in, out)
  barrier
```

data is logically shared by VPs (owner-computes)

Data is eventually gathered accordingly to an user defined way

ASSIST native parmod



An “input section” can be programmed in a CSP-like way

Data items can be distributed (scattered, broadcasted, multicasted) to a set of **Virtual Processes** which are named accordingly to a topology

Data items partitions are elaborated by VPs, possibly in iterative way

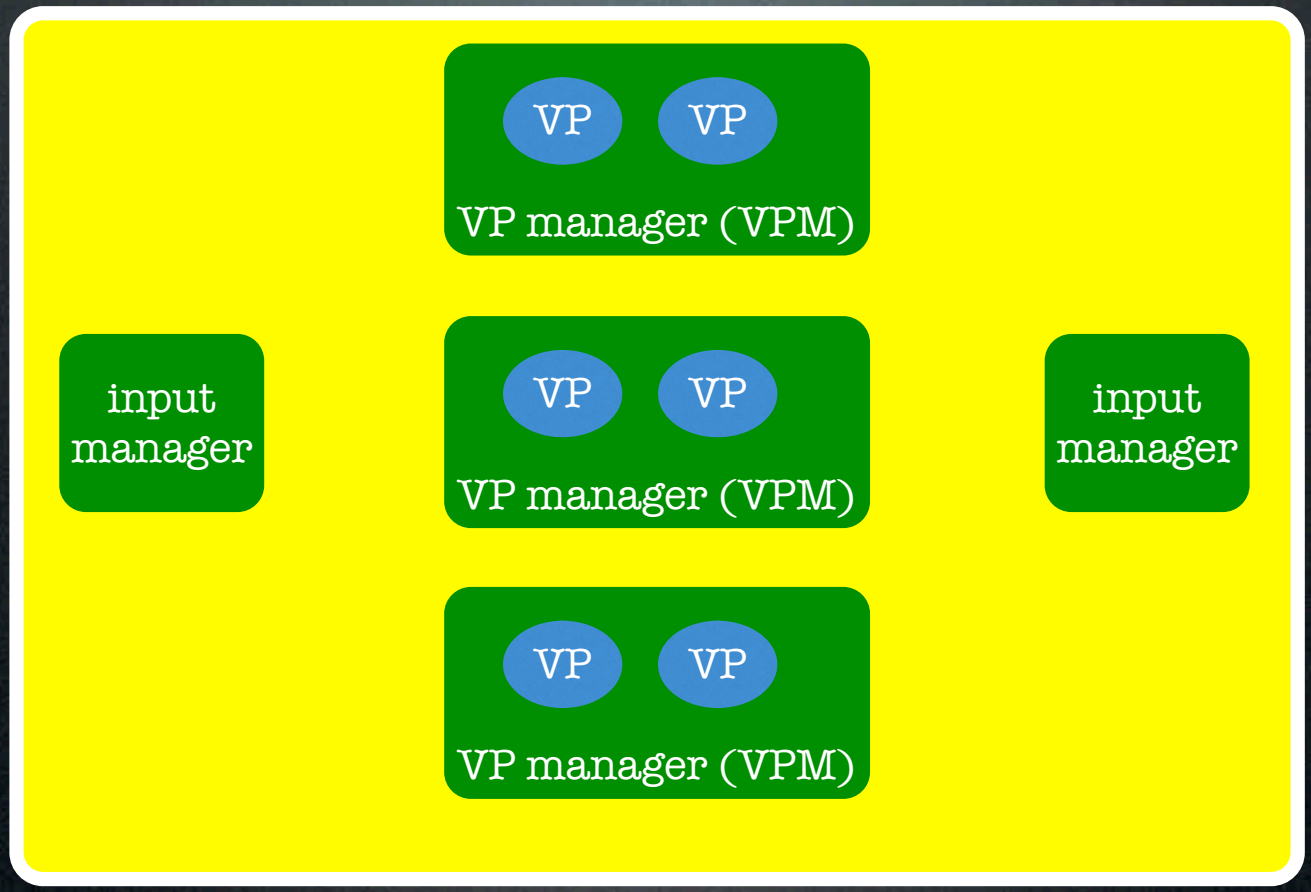
```
while(...)
  forall VP(in, out)
  barrier
```

data is logically shared by VPs (owner-computes)

Data is eventually gathered accordingly to an user defined way

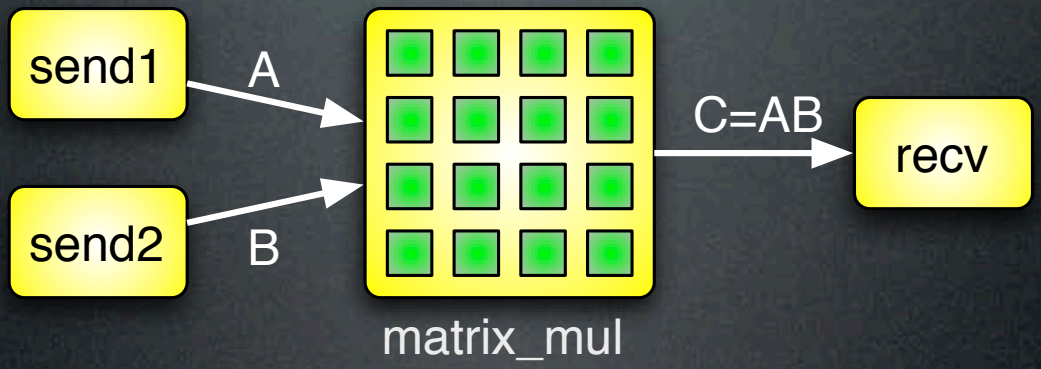
Easy to express standard paradigms (skeltons), such as **farm, deal, haloswap, map, apply-to-all, forall**, ...

parmod implementation



processes VP Virtual Processes

// matrix mul



// matrix mul

```
1  parmod matrix_mul (input_stream long M1[N][N], long M2[N][N]
2                     output_stream long M3[N][N]) {
3    topology array [i:N][j:N] Pv;
4    attribute long A[N][N] scatter A[*ia][*ja] onto Pv[ia][ja];
5    attribute long B[N][N] scatter B[*ib][*jb] onto Pv[ib][jb];
6    stream long ris;
7    do input_section {
8      guard1: on , , M1 && M2 {
9        distribution M1[*i0][*j0] scatter to A[i0][j0];
10       distribution M2[*i1][*j1] scatter to B[i1][j1];
11     }   } while (true)
12   virtual_processes {
13     elab1 (in guard1 out ris) {
14       VP i, j {          f_mul (in A[i][], B[][j] output_stream ris);          }}}
15   output_section {
16     collects ris from ALL Pv[i][j] {
17       int elem; int Matrix_ris_[N][N];
18       AST_FOR_EACH(elem) {
19         Matrix_ris_[i][j]=elem;
20       }
21       assist_out(M3, Matrix_ris_);
22     }<>; } }
23 proc f_mul(in long A[N], long B[N] output_stream long Res)
24 $c++{ register long r=0;
25   for (register int k=0; k<N; ++k)
26     r += A[k]*B[k];
27   assist_out(Res,r); }c++$
```

// matrix mul

```
1   parmod matrix_mul (input_stream long M1[N][N], long M2[N][N]
2                      output_stream long M3[N][N]) {
3     topology array [i:N][j:N] Pv;
4     attribute long A[N][N] scatter A[*ia][*ja] onto Pv[ia][ja];
5     attribute long B[N][N] scatter B[*ib][*jb] onto Pv[ib][jb];
6     stream long ris;
7     do input_section {
8       guard1: on , , M1 && M2 {
9         distribution M1[*i0][*j0] scatter to A[i0][j0];
10        distribution M2[*i1][*j1] scatter to B[i1][j1];
11      }   } while (true)
12    virtual_processes {
13      elab1 (in guard1 out ris) {
14        VP i, j {          f_mul (in A[i][], B[][j] output_stream ris);          }}}
15    output_section {
16      collects ris from ALL Pv[i][j] {
17        int elem; int Matrix_ris_[N][N];
18        AST_FOR_EACH(elem) {
19          Matrix_ris_[i][j]=elem;
20        }
21        assist_out(M3, Matrix_ris_);
22      }<>; } }
23  proc f_mul(in long A[N], long B[N] output_stream long Res)
24  $c++{ register long r=0;
25    for (register int k=0; k<N; ++k)
26      r += A[k]*B[k];
27    assist_out(Res,r); }c++$
```

// matrix mul

```
1   parmod matrix_mul (input_stream long M1[N][N], long M2[N][N]
2                      output_stream long M3[N][N]) {
3     topology array [i:N][j:N] Pv;
4     attribute long A[N][N] scatter A[*ia][*ja] onto Pv[ia][ja];
5     attribute long B[N][N] scatter B[*ib][*jb] onto Pv[ib][jb];
6     stream long ris;
7     do input_section {
8       guard1: on , , M1 && M2 {
9         distribution M1[*i0][*j0] scatter to A[i0][j0];
10        distribution M2[*i1][*j1] scatter to B[i1][j1];
11      }   } while (true)
12    virtual_processes {
13      elab1 (in guard1 out ris) {
14        VP i, j {          f_mul (in A[i][], B[][j] output_stream ris);          }}}
15    output_section {
16      collects ris from ALL Pv[i][j] {
17        int elem; int Matrix_ris_[N][N];
18        AST_FOR_EACH(elem) {
19          Matrix_ris_[i][j]=elem;
20        }
21        assist_out(M3, Matrix_ris_);
22      }<>; } }
23  proc f_mul(in long A[N], long B[N] output_stream long Res)
24  $c++{ register long r=0;
25    for (register int k=0; k<N; ++k)
26      r += A[k]*B[k];
27    assist_out(Res,r); }c++$
```

// matrix mul

```
1   parmod matrix_mul (input_stream long M1[N][N], long M2[N][N]
2                      output_stream long M3[N][N]) {
3     topology array [i:N][j:N] Pv;
4     attribute long A[N][N] scatter A[*ia][*ja] onto Pv[ia][ja];
5     attribute long B[N][N] scatter B[*ib][*jb] onto Pv[ib][jb];
6     stream long ris;
7     do input_section {
8       guard1: on , , M1 && M2 {
9         distribution M1[*i0][*j0] scatter to A[i0][j0];
10        distribution M2[*i1][*j1] scatter to B[i1][j1];
11      }   } while (true)
12    virtual_processes {
13      elab1 (in guard1 out ris) {
14        VP i, j {          f_mul (in A[i][], B[][j] output_stream ris);          }}}
15    output_section {
16      collects ris from ALL Pv[i][j] {
17        int elem; int Matrix_ris_[N][N];
18        AST_FOR_EACH(elem) {
19          Matrix_ris_[i][j]=elem;
20        }
21        assist_out(M3, Matrix_ris_);
22      }<>; } }
23  proc f_mul(in long A[N], long B[N] output_stream long Res)
24  $c++{ register long r=0;
25    for (register int k=0; k<N; ++k)
26      r += A[k]*B[k];
27    assist_out(Res,r); }c++$
```

// matrix mul

```
1   parmod matrix_mul (input_stream long M1[N][N], long M2[N][N]
2                      output_stream long M3[N][N]) {
3     topology array [i:N][j:N] Pv;
4     attribute long A[N][N] scatter A[*ia][*ja] onto Pv[ia][ja];
5     attribute long B[N][N] scatter B[*ib][*jb] onto Pv[ib][jb];
6     stream long ris;
7     do input_section {
8       guard1: on , , M1 && M2 {
9         distribution M1[*i0][*j0] scatter to A[i0][j0];
10        distribution M2[*i1][*j1] scatter to B[i1][j1];
11      }   } while (true)
12    virtual_processes {
13      elab1 (in guard1 out ris) {
14        VP i, j {          f_mul (in A[i][], B[][j] output_stream ris);          }}}
15    output_section {
16      collects ris from ALL Pv[i][j] {
17        int elem; int Matrix_ris_[N][N];
18        AST_FOR_EACH(elem) {
19          Matrix_ris_[i][j]=elem;
20        }
21        assist_out(M3, Matrix_ris_);
22      }<>; } }
23  proc f_mul(in long A[N], long B[N] output_stream long Res)
24  $c++{ register long r=0;
25    for (register int k=0; k<N; ++k)
26      r += A[k]*B[k];
27    assist_out(Res,r); }c++$
```

// matrix mul

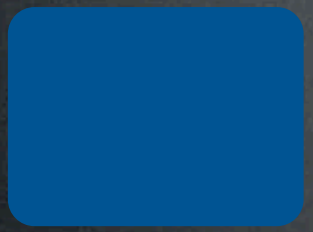
```
1   parmod matrix_mul (input_stream long M1[N][N], long M2[N][N]
2                      output_stream long M3[N][N]) {
3     topology array [i:N][j:N] Pv;
4     attribute long A[N][N] scatter A[*ia][*ja] onto Pv[ia][ja];
5     attribute long B[N][N] scatter B[*ib][*jb] onto Pv[ib][jb];
6     stream long ris;
7     do input_section {
8       guard1: on , , M1 && M2 {
9         distribution M1[*i0][*j0] scatter to A[i0][j0];
10        distribution M2[*i1][*j1] scatter to B[i1][j1];
11      }   } while (true)
12    virtual_processes {
13      elab1 (in guard1 out ris) {
14        VP i, j {          f_mul (in A[i][], B[][j] output_stream ris);          }}}
15    output_section {
16      collects ris from ALL Pv[i][j] {
17        int elem; int Matrix_ris_[N][N];
18        AST_FOR_EACH(elem) {
19          Matrix_ris_[i][j]=elem;
20        }
21        assist_out(M3, Matrix_ris_);
22      }<>; } }
23  proc f_mul(in long A[N], long B[N] output_stream long Res)
24  $c++{ register long r=0;
25    for (register int k=0; k<N; ++k)
26      r += A[k]*B[k];
27    assist_out(Res,r); }c++$
```

// matrix mul

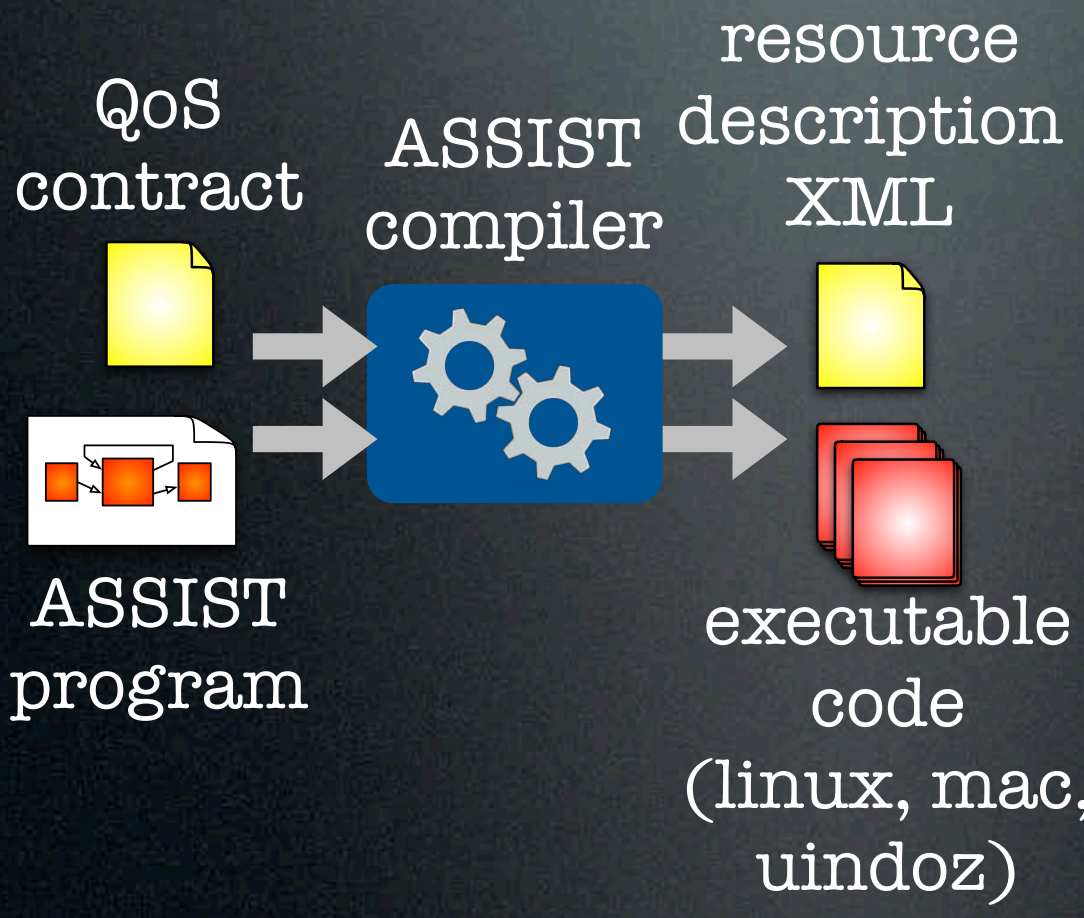
```
1  parmod matrix_mul (input_stream long M1[N][N], long M2[N][N]
2                     output_stream long M3[N][N]) {
3    topology array [i:N][j:N] Pv;
4    attribute long A[N][N] scatter A[*ia][*ja] onto Pv[ia][ja];
5    attribute long B[N][N] scatter B[*ib][*jb] onto Pv[ib][jb];
6    stream long ris;
7    do input_section {
8      guard1: on , , M1 && M2 {
9        distribution M1[*i0][*j0] scatter to A[i0][j0];
10       distribution M2[*i1][*j1] scatter to B[i1][j1];
11     }   } while (true)
12   virtual_processes {
13     elab1 (in guard1 out ris) {
14       VP i, j {          f_mul (in A[i][], B[][j] output_stream ris);          }}}
15   output_section {
16     collects ris from ALL Pv[i][j] {
17       int elem; int Matrix_ris_[N][N];
18       AST_FOR_EACH(elem) {
19         Matrix_ris_[i][j]=elem;
20       }
21       assist_out(M3, Matrix_ris_);
22     }<>; } }
23 proc f_mul(in long A[N], long B[N] output_stream long Res)
24 $c++{ register long r=0;
25   for (register int k=0; k<N; ++k)
26     r += A[k]*B[k];
27   assist_out(Res,r); }c++$
```

Compiling & Running

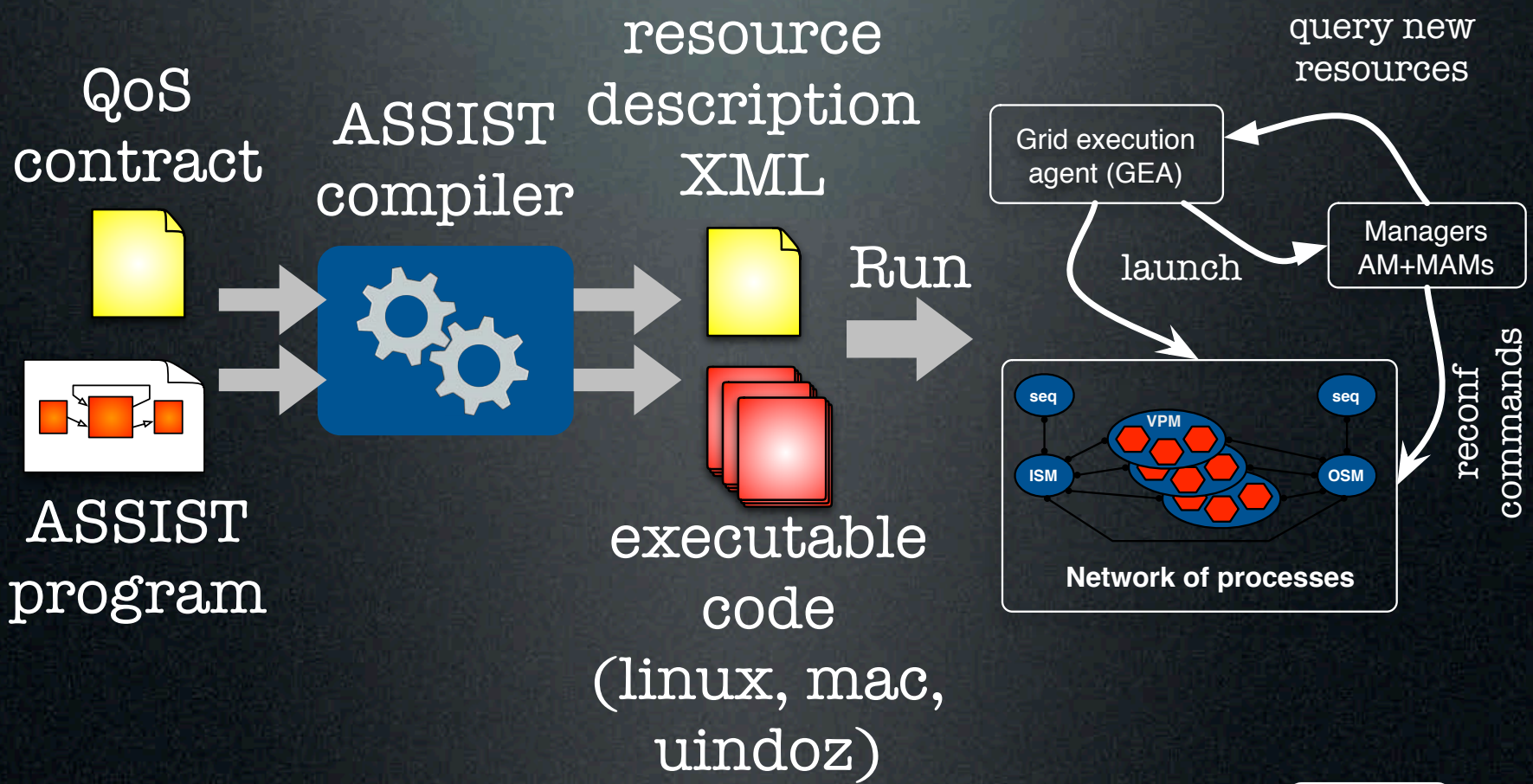
ASSIST
compiler



Compiling & Running



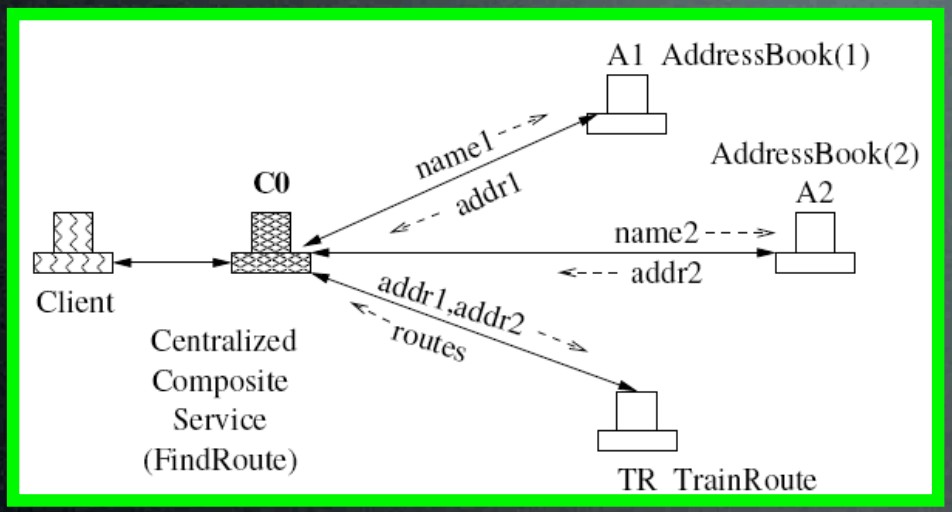
Compiling & Running



WS as transport

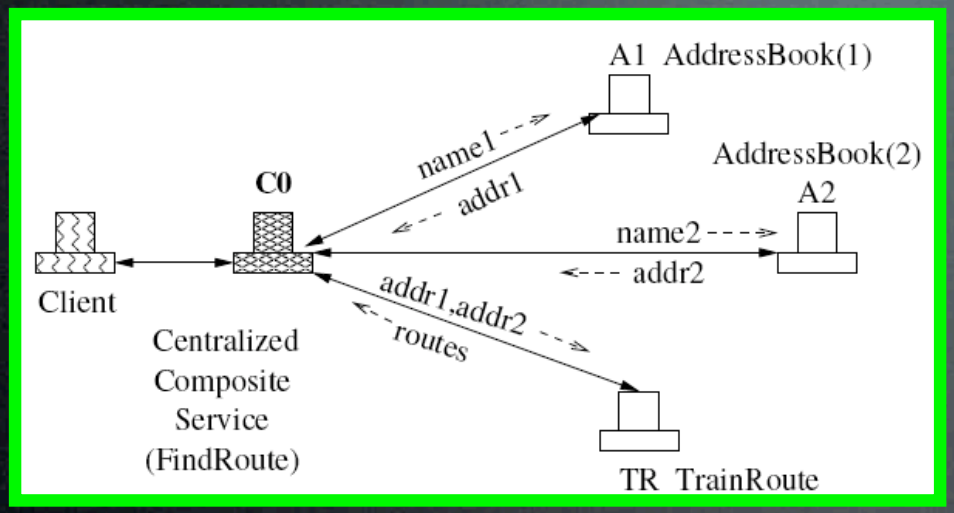
- ASSIST can use WS as transport for streams
- provide interoperability with standards
  - automatically generated
  - helps in dealing with firewalls
- on the whole, an ASSIST app with WS can be considered a composite service with distributed orchestration

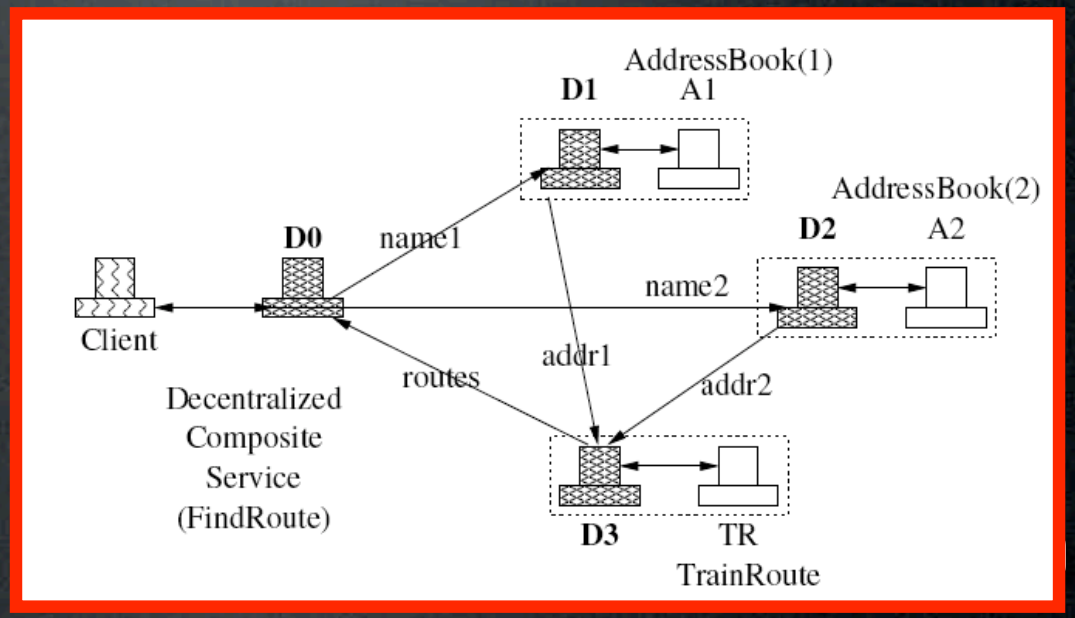
BPEL distributed orchestration



Nanda et al. (IBM)
Decentralizing Execution
of Composite Web Services
OOPSLA 2004

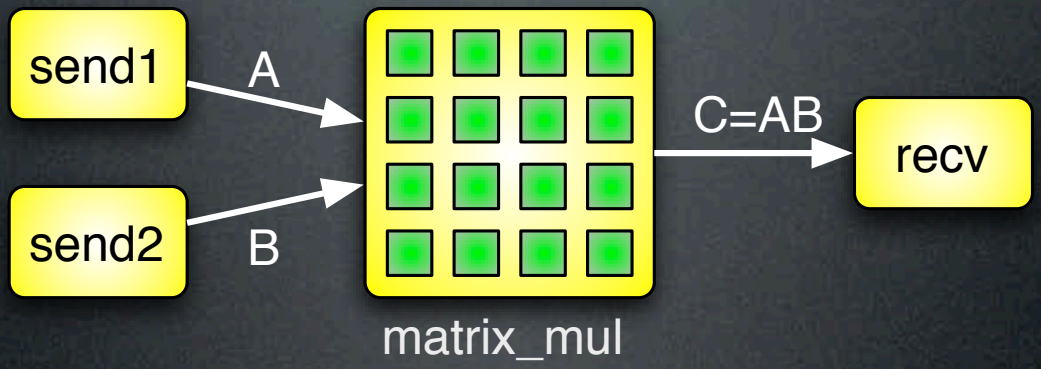
BPEL distributed orchestration





Nanda et al. (IBM)
Decentralizing Execution
of Composite Web Services
OOPSLA 2004

// matrix mul



WS composition

```
<ComponentConfiguration>
 <Assembly>
  <ComponentSection>
   <Component name="send1"  com="ws" kind="xml" file="./xmls/send1.xml"> </Component>
   <Component name="send2"  com="ws" kind="xml" file="./xmls/send2.xml"> </Component>
   <Component name="matrix_mul"  com="ws" kind="xml" file="./xmls/matrix_mul.xml"> </Compo
   <Component name="recv"  com="ws" kind="xml" file="./xmls/rec.xml"> </Component>
  </ComponentSection>
  <ConnectionSection>
   <Connection>
    <Output component="send1" interface="Matrix1"/>
    <Input component="matrix_mul" interface="Matrix1"/>
   </Connection>
   <Connection>
    <Output component="send2" interface="Matrix2"/>
    <Input component="matrix_mul" interface="Matrix2"/>
   </Connection>
   <Connection>
    <Output component="matrix_mul" interface="Matrix_ris"/>
    <Input component="recv" interface="Matrix"/>
   </Connection>
  </ConnectionSection>
 </Assembly>
</ComponentConfiguration>
```

WS composition

```
<ComponentConfiguration>
 <Assembly>
  <ComponentSection>
   <Component name="send1"  com="ws" kind="xml" file="./xmls/send1.xml"> </Component>
   <Component name="send2"  com="ws" kind="xml" file="./xmls/send2.xml"> </Component>
   <Component name="matrix_mul"  com="ws" kind="xml" file="./xmls/matrix_mul.xml"> </Compo
   <Component name="recv"  com="ws" kind="xml" file="./xmls/rec.xml"> </Component>
  </ComponentSection>
  <ConnectionSection>
   <Connection>
    <Output component="send1" interface="Matrix1"/>
    <Input component="matrix_mul" interface="Matrix1"/>
   </Connection>
   <Connection>
    <Output component="send2" interface="Matrix2"/>
    <Input component="matrix_mul" interface="Matrix2"/>
   </Connection>
   <Connection>
    <Output component="matrix_mul" interface="Matrix_ris"/>
    <Input component="recv" interface="Matrix"/>
   </Connection>
  </ConnectionSection>
 </Assembly>
</ComponentConfiguration>
```

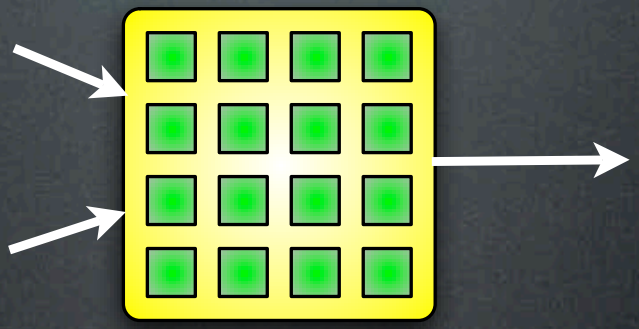
WS composition

```
<ComponentConfiguration>
 <Assembly>
  <ComponentSection>
   <Component name="send1"  com="ws" kind="xml" file="./xmls/send1.xml"> </Component>
   <Component name="send2"  com="ws" kind="xml" file="./xmls/send2.xml"> </Component>
   <Component name="matrix_mul"  com="ws" kind="xml" file="./xmls/matrix_mul.xml"> </Compo
   <Component name="recv"  com="ws" kind="xml" file="./xmls/rec.xml"> </Component>
  </ComponentSection>
  <ConnectionSection>
   <Connection>
    <Output component="send1" interface="Matrix1"/>
    <Input component="matrix_mul" interface="Matrix1"/>
   </Connection>
   <Connection>
    <Output component="send2" interface="Matrix2"/>
    <Input component="matrix_mul" interface="Matrix2"/>
   </Connection>
   <Connection>
    <Output component="matrix_mul" interface="Matrix_ris"/>
    <Input component="recv" interface="Matrix"/>
   </Connection>
  </ConnectionSection>
 </Assembly>
</ComponentConfiguration>
```

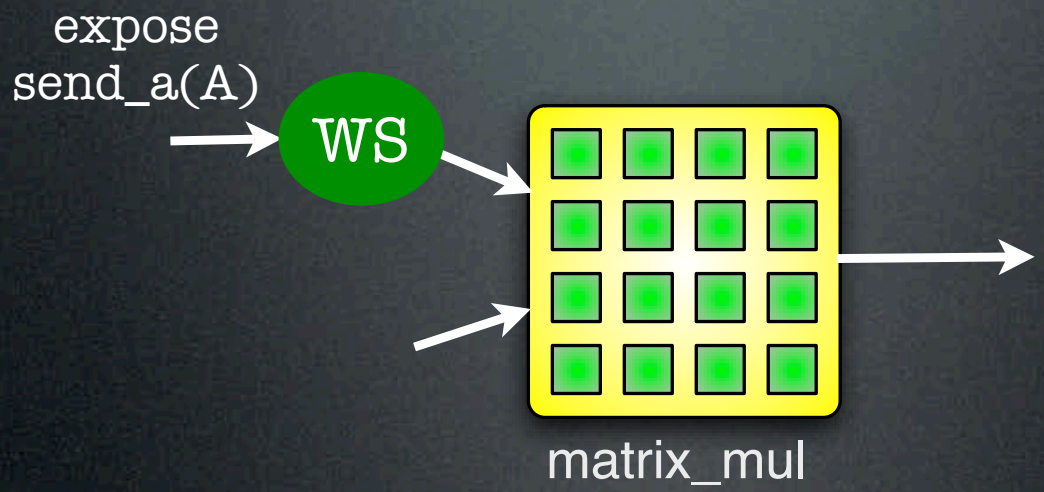
WS composition

```
<ComponentConfiguration>
 <Assembly>
  <ComponentSection>
   <Component name="send1"  com="ws" kind="xml" file="./xmls/send1.xml"> </Component>
   <Component name="send2"  com="ws" kind="xml" file="./xmls/send2.xml"> </Component>
   <Component name="matrix_mul"  com="ws" kind="xml" file="./xmls/matrix_mul.xml"> </Compo
   <Component name="recv"  com="ws" kind="xml" file="./xmls/rec.xml"> </Component>
  </ComponentSection>
  <ConnectionSection>
   <Connection>
    <Output component="send1" interface="Matrix1"/>
    <Input component="matrix_mul" interface="Matrix1"/>
   </Connection>
   <Connection>
    <Output component="send2" interface="Matrix2"/>
    <Input component="matrix_mul" interface="Matrix2"/>
   </Connection>
   <Connection>
    <Output component="matrix_mul" interface="Matrix_ris"/>
    <Input component="recv" interface="Matrix"/>
   </Connection>
  </ConnectionSection>
 </Assembly>
</ComponentConfiguration>
```

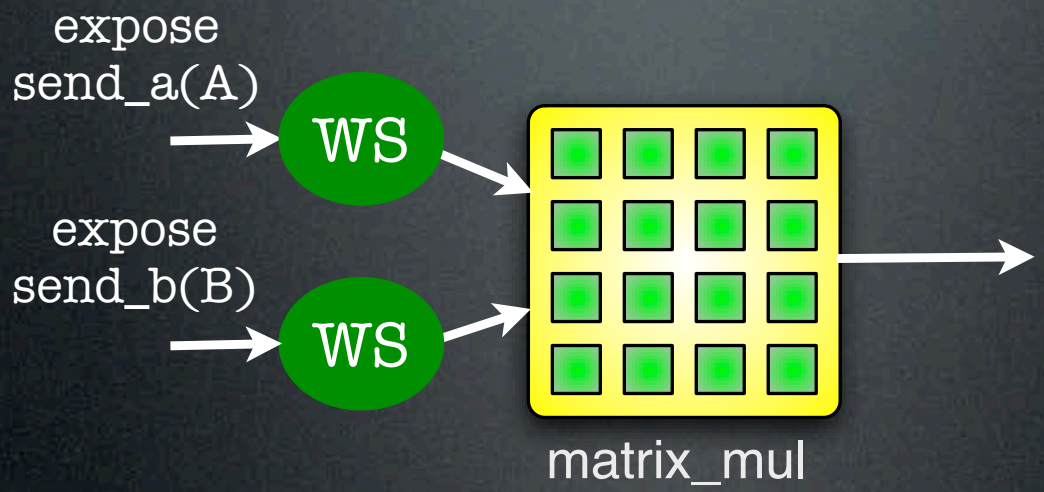
Generating WS



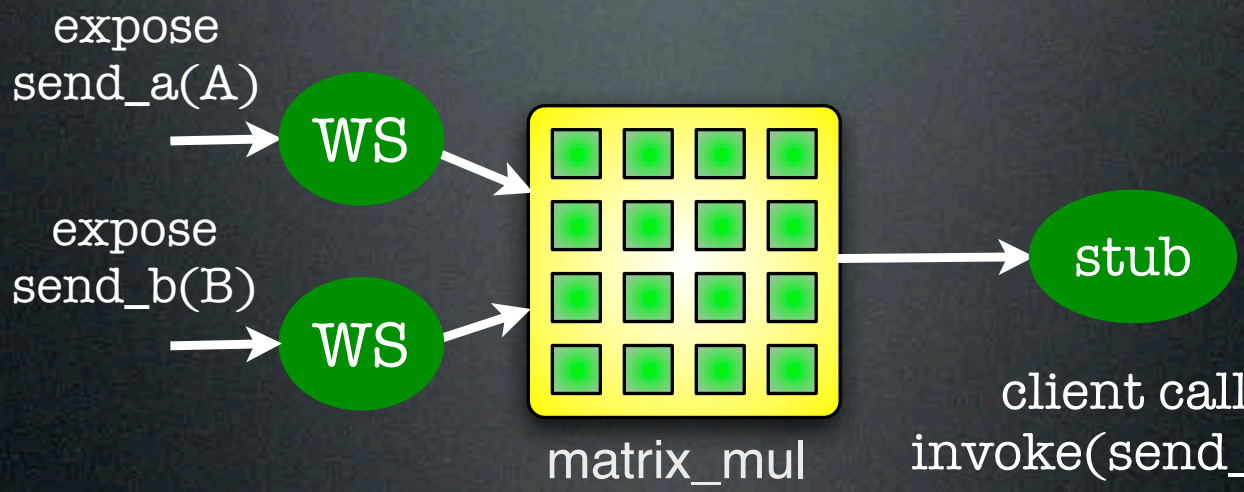
Generating WS



Generating WS



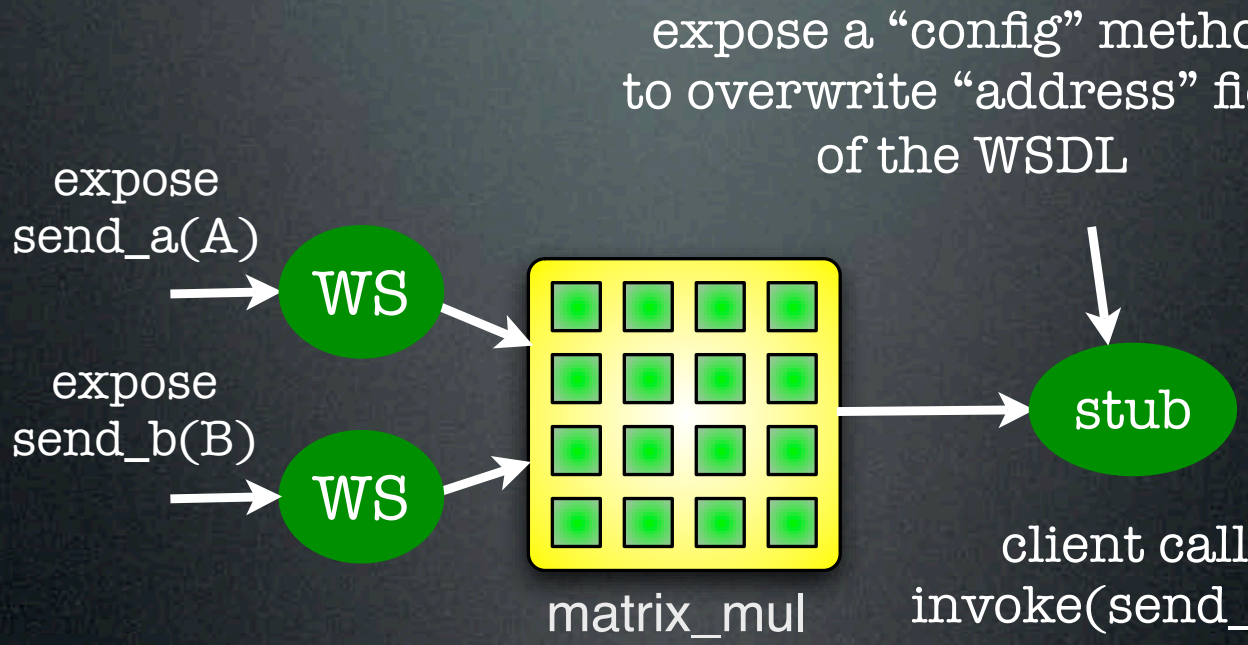
Generating WS



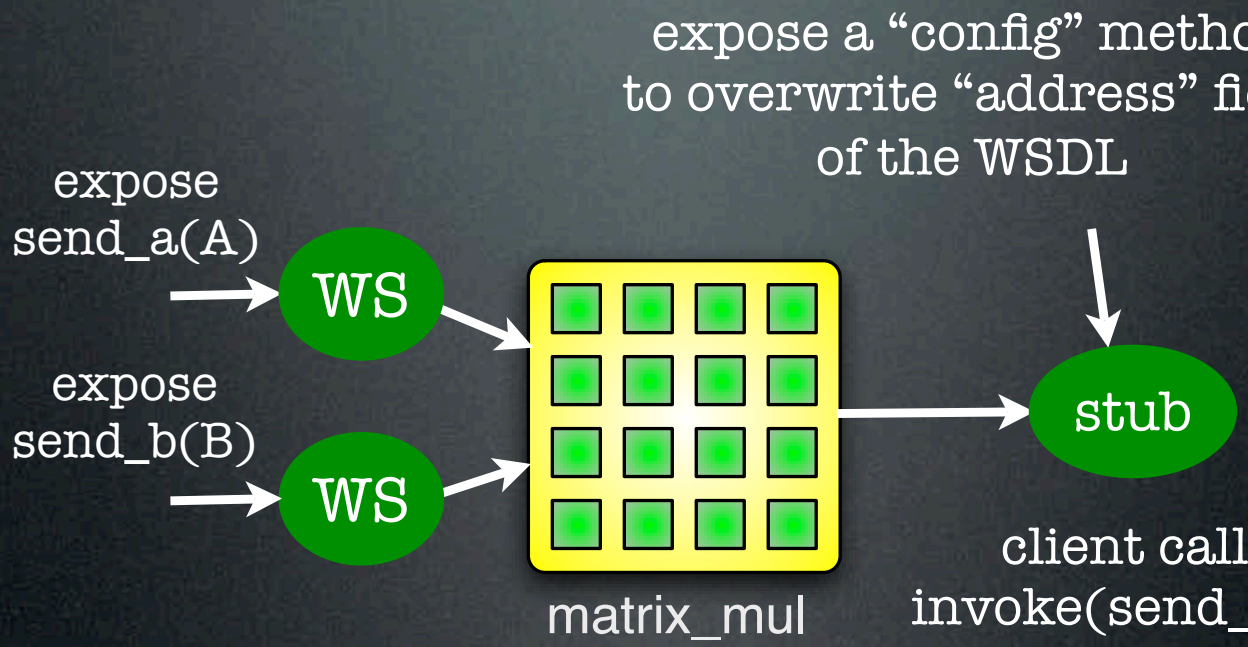
client calling
invoke(send_x(X)),
where X is decided
at wiring time

generated at wiring time
starting from the WSDL of X

Generating WS



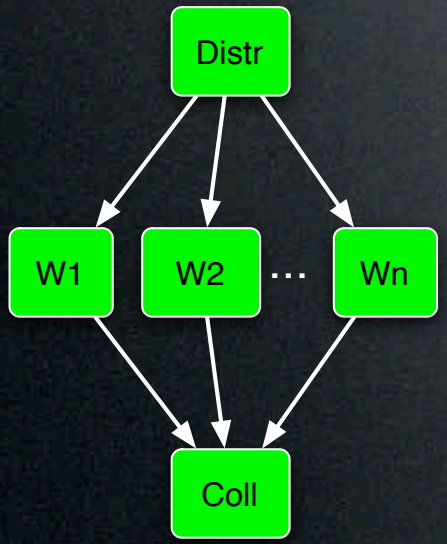
Generating WS



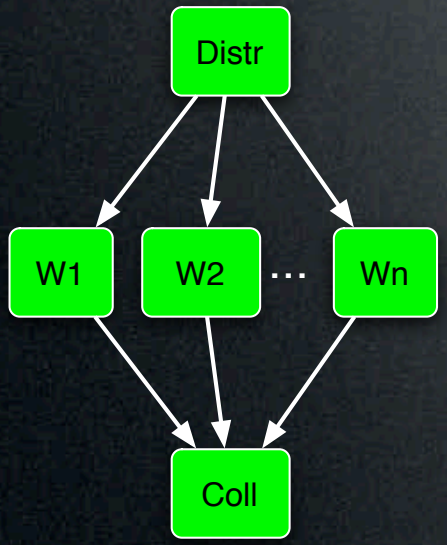
WS are implemented by means of gSOAP embedded server

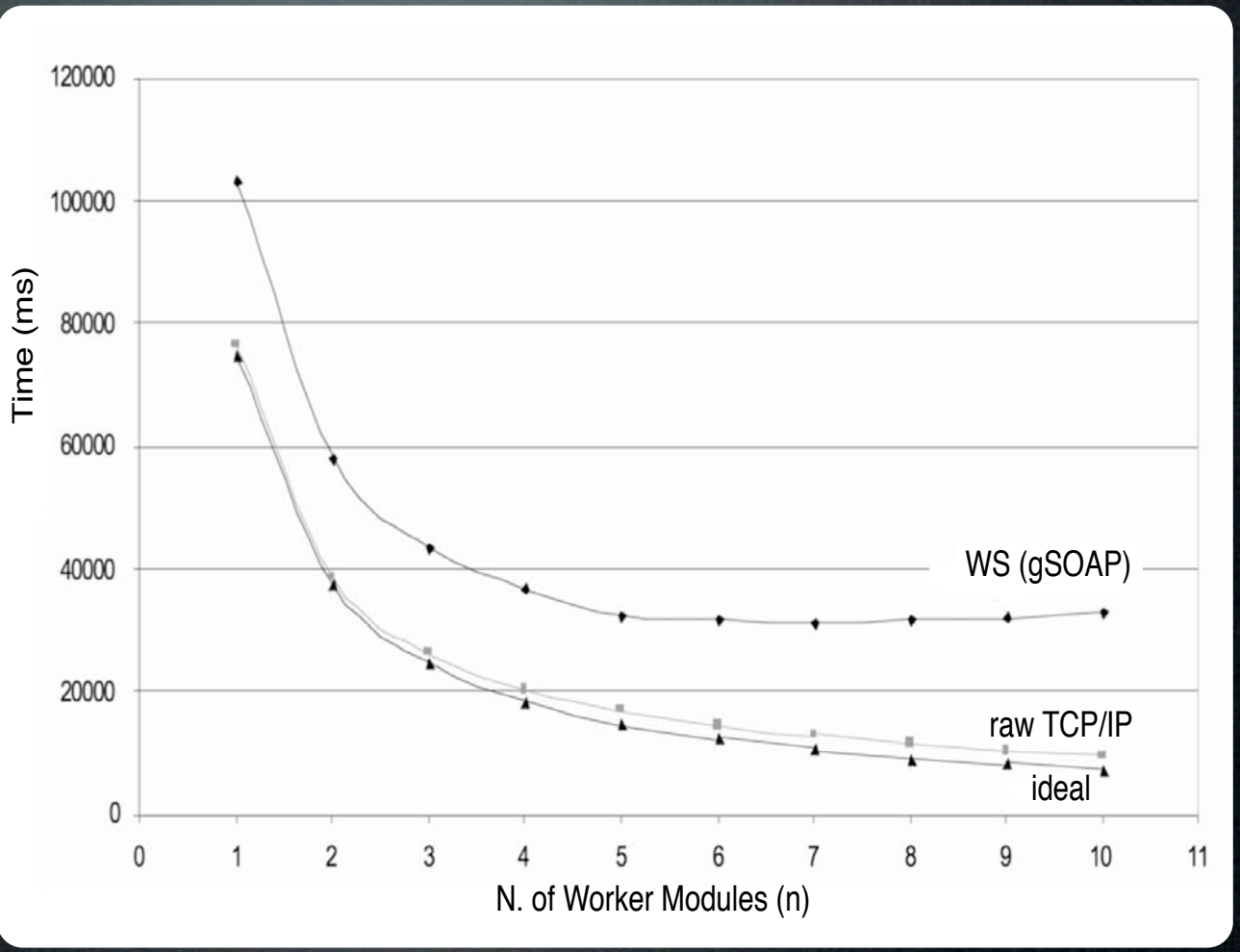
generated at wiring time starting from the WSDL of X

A simple farm with WS

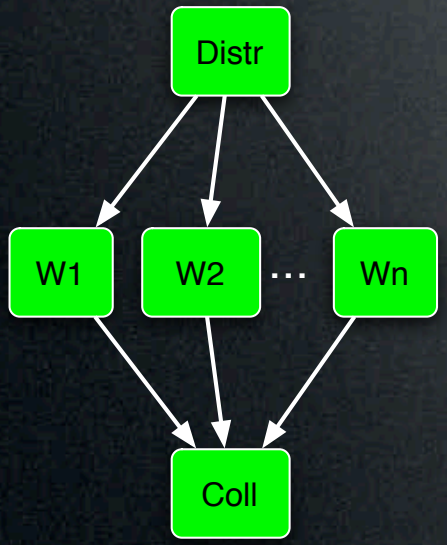


A simple farm with WS

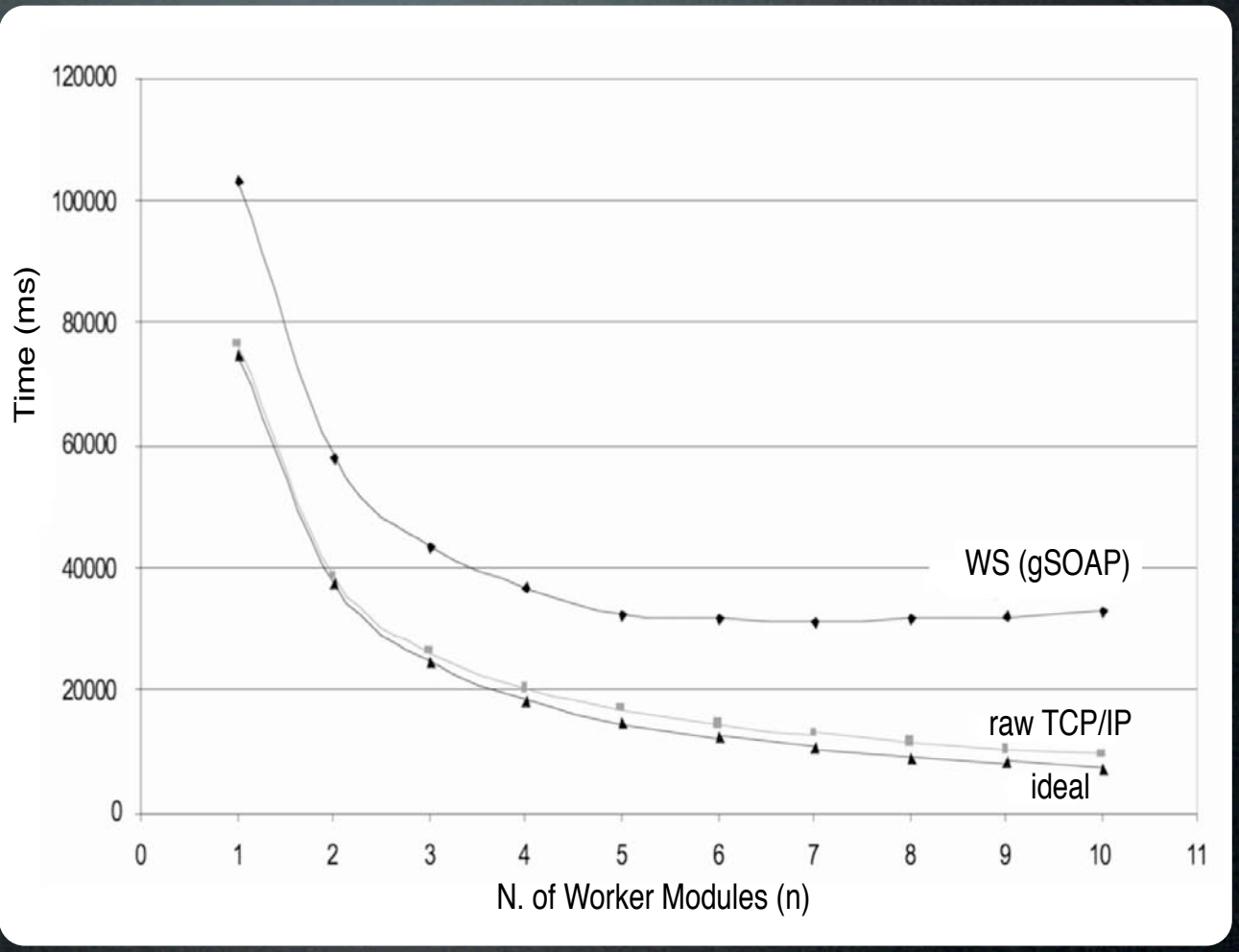




A simple farm with WS



30-50% communication slowdown



Composite WS & ASSIST

- Differently from BPEL, we start from the graph
- not a big deal, however not fully compliant with pragmatic approach to WS
  - we also would like to provide full interoperability with RPC WS

ASSIST & components

- moving towards component approach
- an ASSIST module of a graph of them may be defined as Grid.it component
  - stream ports (use/provide)
  - RPC ports (use/provide)
- component wiring through ASSIST native, HTTP/SOAP, IIOP/CORBA (for CCM components)

By the way ...

By the way ...

- is a WS a component?
  - following the Szyperski's definition

By the way ...

- is a WS a component?
  - following the Szyperski's definition
- Dennis Gannon said yes!
  - Europar 2004 invited talk
  - we also believe that

Euro-Par, Pisa 2004

BUILDING GRID APPLICATIONS AND PORTALS:
An Approach Based on Components, Web Services and Workflow Tools.

D. Gannon, B. Plale
Ph.D. Students: S. Krishnan, L. Fang, G. Kandaswamy, Y. Simmhan,
A. Slominski
Indiana University

CCA components as Web Services

- A natural extension of the model:
  - Each Provides port can be a complete web service
  - Uses ports become web service "client stubs".
- A Connection is then a binding between a client stub and the WSDL for the some provides port.
- XCAT3 implements this feature.
  - Uses python as the scripting language.
- Also based on the OGSI standard.

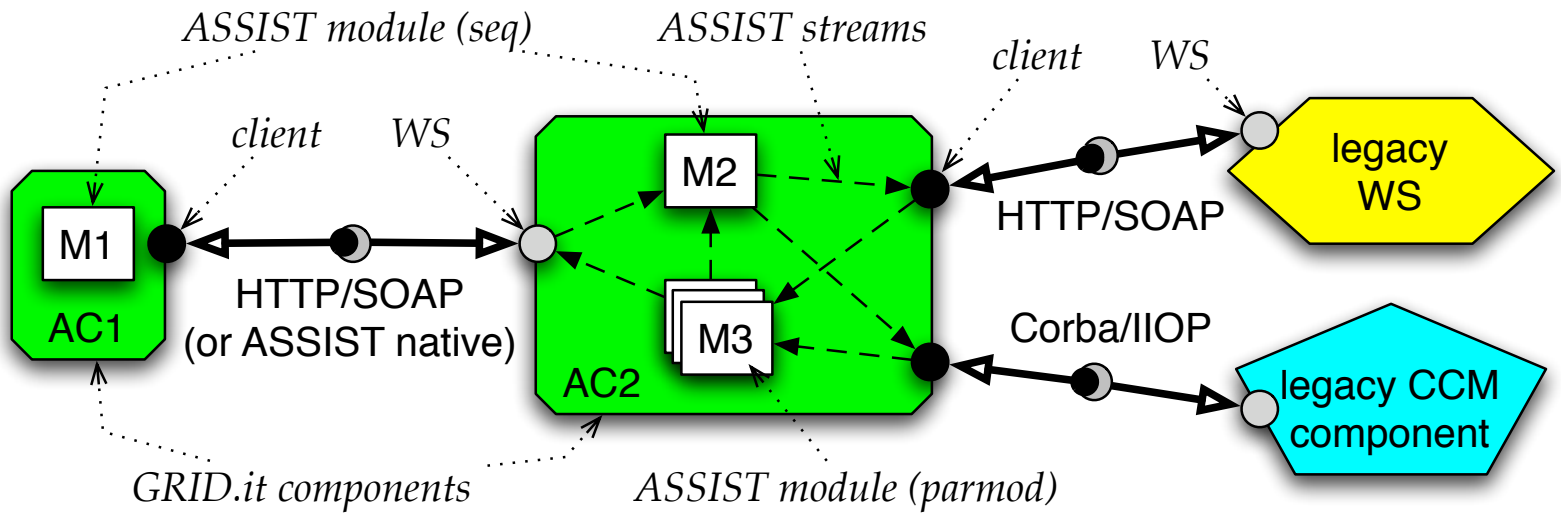
Web Services as CCA components

- Can we use the Google web service as a CCA component?
- Message oriented and not RCP based.
  - Send a message to the service
    - You may get a response or you may not.
      - Depends upon the service semantics.
- No concept of "uses port".
  - However some serves generate messages in response to messages sent.
  - Web service addressing allows a reply to be forwarded to a "3rd party" receiver.

Predicting Severe Storms

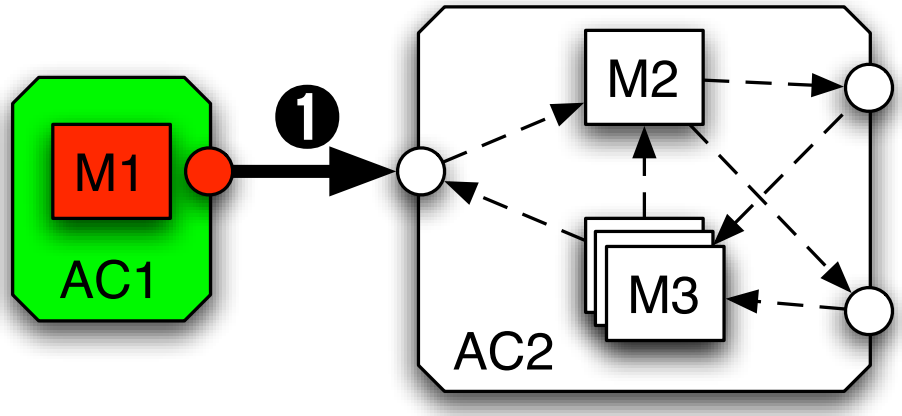
- To deliver better than real-time predictions
  - Data mining of live instrument streams and historical storm metadata
  - Requisition large computational resources on demand to start a large number of simulations
    - Mine simulation outputs to see which track real storm evolution.
    - Refine scenarios that match incoming data.
  - May Need to requisition bandwidth to make the needed data analysis possible.
  - May require real-time re-alignment of instruments.
  - Workflows may run for a long time and they must be adaptive and very dynamic

Components: the big picture



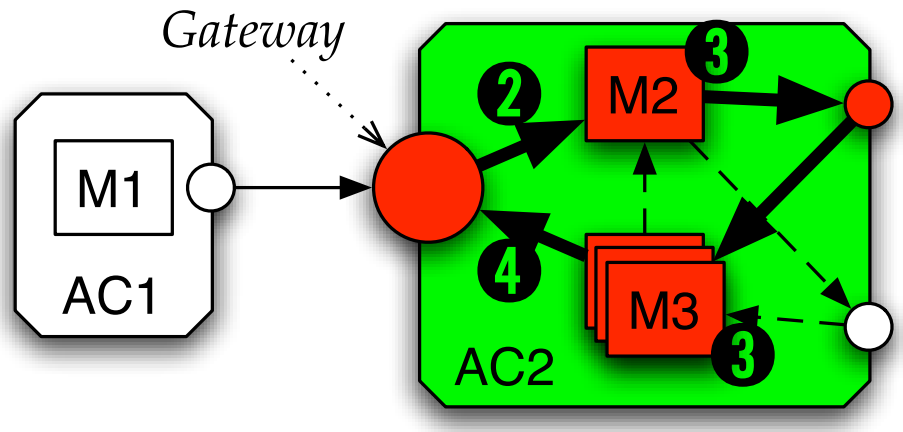
Components: the big picture

❶ A method invocation arrives to RPC provide port of AC2



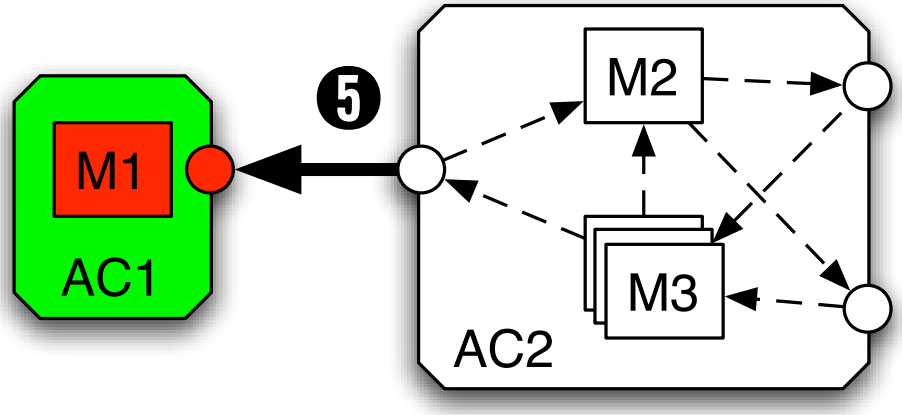
Components: the big picture

1. A method invocation arrives to RPC provide port of AC2
2. Parameters are injected in a input stream together with an unique id
3. Computation is performed
4. As soon as a matching id arrive to the same port a reply message is prepared



Components: the big picture

1. A method invocation arrives to RPC provide port of AC2
2. Parameters are injected in a input stream together with an unique id
3. Computation is performed
4. As soon as a matching id arrive to the same port a reply message is prepared
5. The method invocation is finalized with a result message



Conclusions

- WS extension is not rocket science, but being compliant to standards may make the difference for real applications
- ASSIST provide high-level programming for grid
  - dynamic adaptivity, autonomic QoS control, fault-tolerance (ongoing), ... and this is rocket science
- interoperability with WS & CCM
  - transparent to the programmer, automatic generation of the needed adaptors
  - distributed orchestration of workflows
  - the run-time exploits standard middleware (POSIX/TCP, Globus, ...), it provide to the programmer higher-level view of it

Thank you

The ASSIST programming toolkit has been designed at the University of Pisa, Italy ...