Grid and Component Technologies in Physics Applications

Svetlana Shasharina

Nanbor Wang, Stefan Muszala and Roopa Pundaleeka.

sveta@txcorp.com

Tech-X Corporation

Boulder, CO

ICALEPCS07, October 15, 2007

Outline - our experience in bridging computer science and physics

- Company Intro

- Grids
- Fusion Grid Service

- Components and CCA
- Babel
- Components in Fusion Simulation Project

Disclaimer

- All definitions are not strict and rather personal
- Opinions are based on our experience in bringing Grids and Components into physics applications
- Check with the official web sites and papers for rigorous definitions of the technologies and their success stories

Tech-X Corporation (www.txcorp.com) bridges physics and computer science

- **Founded in 1994, now ~50 people**
- **Accelerator physics (traditional and advanced)**
- **Fusion theory**
- **Plasma modeling**
- **Computer Science**
  - C++, Java, Python
  - Grids, Globus
  - Web Services
  - CORBA
  - XML
  - Visualization
  - Data management
  - Workflows
  - Components
  - Mission-critical systems (Real Tim
- **Small Business Innovation Research (DOE, NASA, DOD)**
- **SciDAC program**
  - Lead in FACETS
  - Significant in COMPASS
  - TASCS (CCA)

Grids are aggregated and secured resources + ???

- **Grids are computers connected by network (WAN, otherwise we deal with cluster or super computing - not our subject).**
- **This aggregation of resources is used by a certain group (Virtual Organization) so that more resources are available to each member and the outsiders are separated.**
  - Needs for certificates (authentication and authorization)
- **If this aggregation is just a sum of constituents, it is not valuable**
  - Why do I need to install Globus, learn how to use it, get and update my certificate if I can login to any machine, run a job and scp my data?
- **Grid should add value, hence one needs middleware that adds value by providing default services and allows adding new ones**

Services make Grids

- **Services do anything not trivial on distributed resources**
  - Some are provided by Grid software (for example, file transfer and job managing services of Globus)
  - Some can be built using Grid software (example is coming)
- **Services are implementations of abstractions so that the resources are virtualized**
  - Data represented by an attribute (give me all files with current larger than x) or computers represented by baselines (run my job anywhere as long as the node is configured right)
- **Services can:**
  - Save storage (no need to replicate data)
  - Save travel funds (no need to travel to access resources)
  - Allow new types of research (global data and global network analysis and optimization)
  - Create complex data access and distribution system needed for large experiments like LHC and ITER
- **Grid - many resources for many users securely using for something non-trivial**

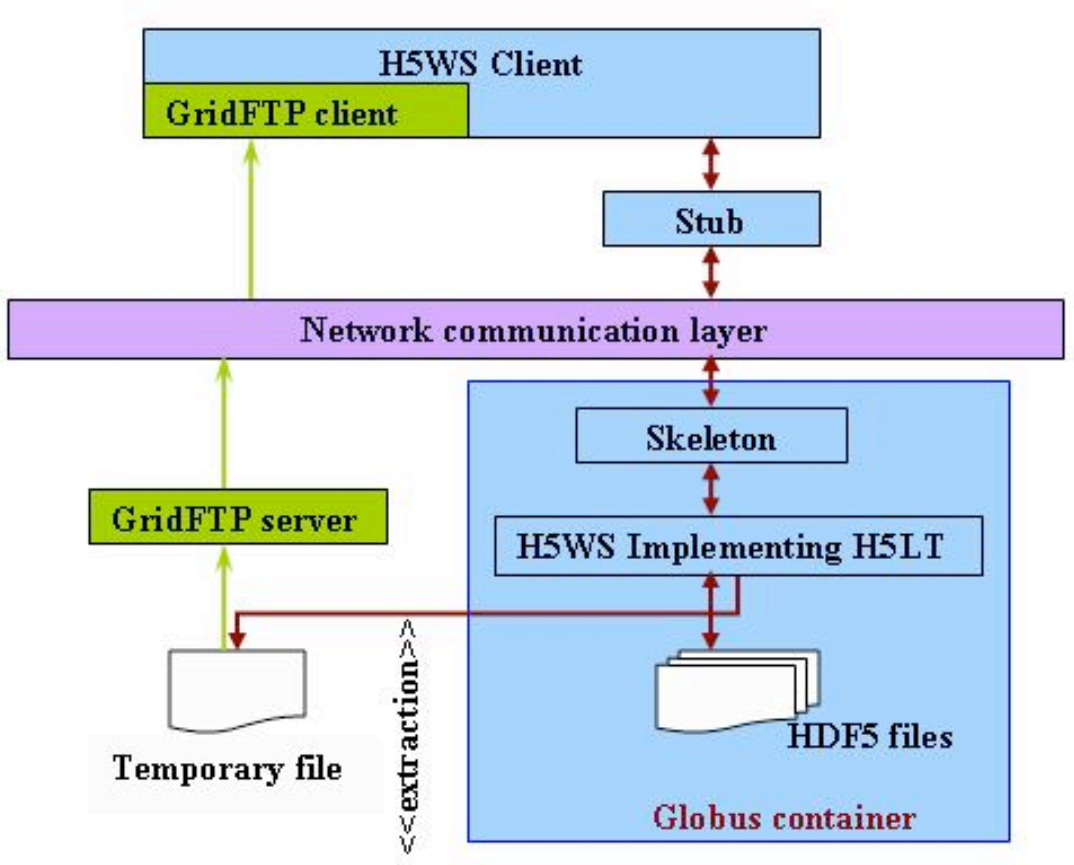
Fusion Grid Service (H5WS) is an example of an service built using Globus to provide services for fusion community

- **Motivation**
  - Data is generated remotely but needs to be analyzed locally
  - Only subsets of data are needed
  - HDF5 is popular (file format and API) but works on local files
  - Hence the goal is a service
    - With methods mimicking HDF5 API
    - Bringing data into client memory with optional caching
    - Efficient data transfers (not SOAP)
- **Desired API**
  - Query dataset for its metadata (rank, dimensions and type)
  - Bring a dataset
  - Bring a hyberslab (a cube with specified stride)
  - Bring a whole file
- **Globus (4.0.2) was chosen as technology**
  - GridFTP
  - C bindings (for using with HDF5 C API)
  - Political favorite at the time

Start service development by defining interface (types and functions) in WSDL

- **User-defined data type for dataset metadata**
  - Using integers (for type and rank) and sequence types (for dimensions)
- **Get metadata**
  - Using WSDL native types and our user-defined type define a functions
  - Metadata is obtained through a SOAP call
- **Need something else (not SOAP) for data itself so the method (in the client wrapper) is composite:**
  - Treat all data as 1D array (reshape as needed)
  - Ask for the metadata
  - Extract data on the server side into a file (do not know how to do from-memory-to-memory transfers)
  - Allocate the memory on the client
  - Use GridFTP C API to dump the file into the client memory

H5WS has C++ client wrapper for abstraction and delegation to GridFTP and server delegating to HDF5 API



We compared performance with other technologies

- **Our H5WS (based on Globus Web Service with GridFTP)**
- **CORBA (TAO-1.5.2)**
  - Allows binary data type (octet)
  - C++ bindings
- **gSoap-2.7**
  - Allows DIME attachments
  - C++ bindings
- **Performance tests:**
  - Establish connection
  - Get metadata
  - Allocate memory
  - Extract dataset (with extra step of temporary file for Globus)
  - Close connection

Three test settings of typical use

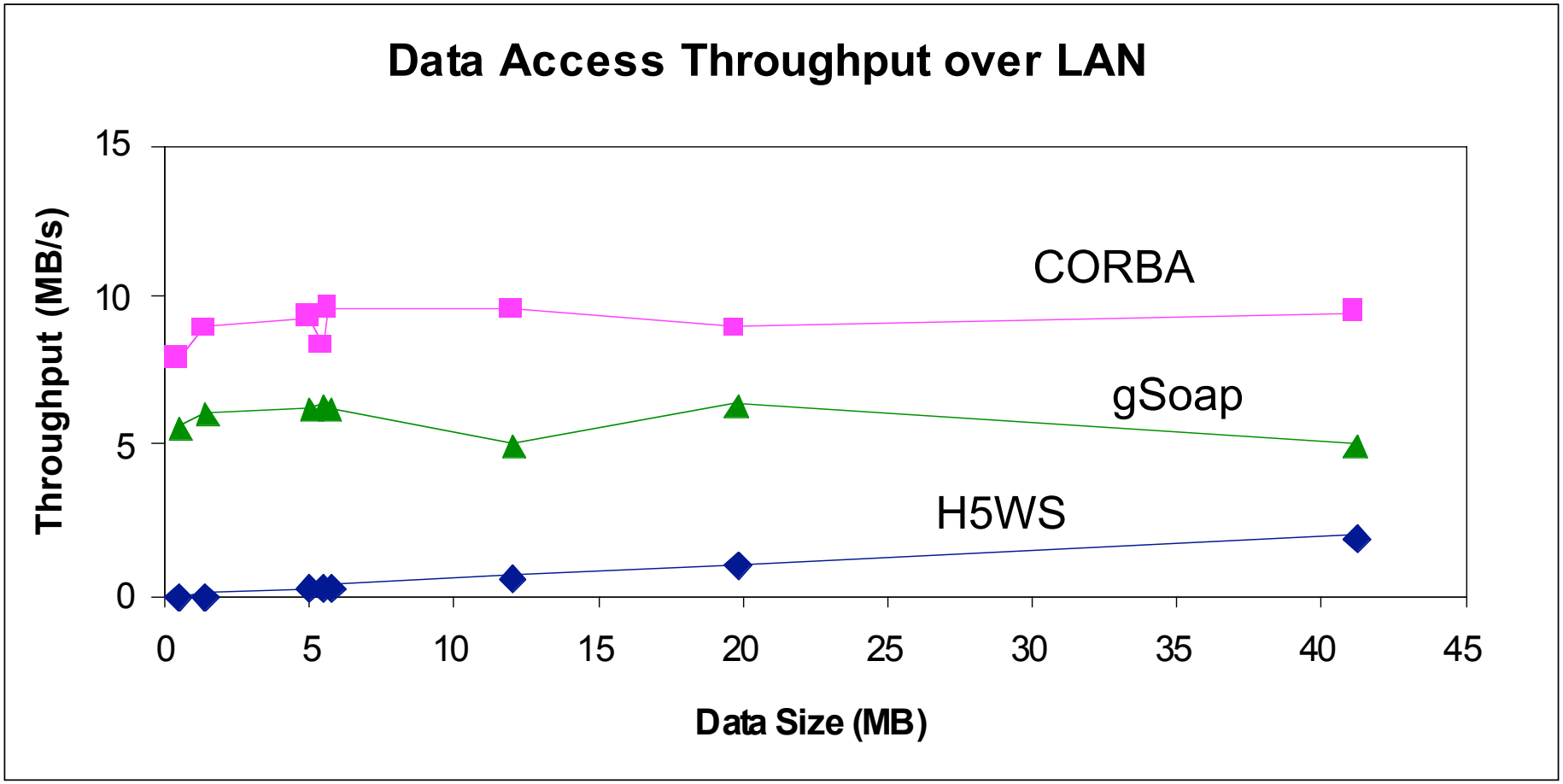
- LAN - 2 Tech-X Linus boxes (grid and storage2)
- ESnet - NERSC (davinci) and PPPL (grid0)
- WAN - NERSC to Tech-X

Setup	Bottleneck bandwidth (Mbyte/sec)	RTT (msec)	BDP
LAN	12.5	0.27	3.4 KB
ESnet	125.0	72.0	9.0 MB
WAN	0.19	162.0	31.4 KB

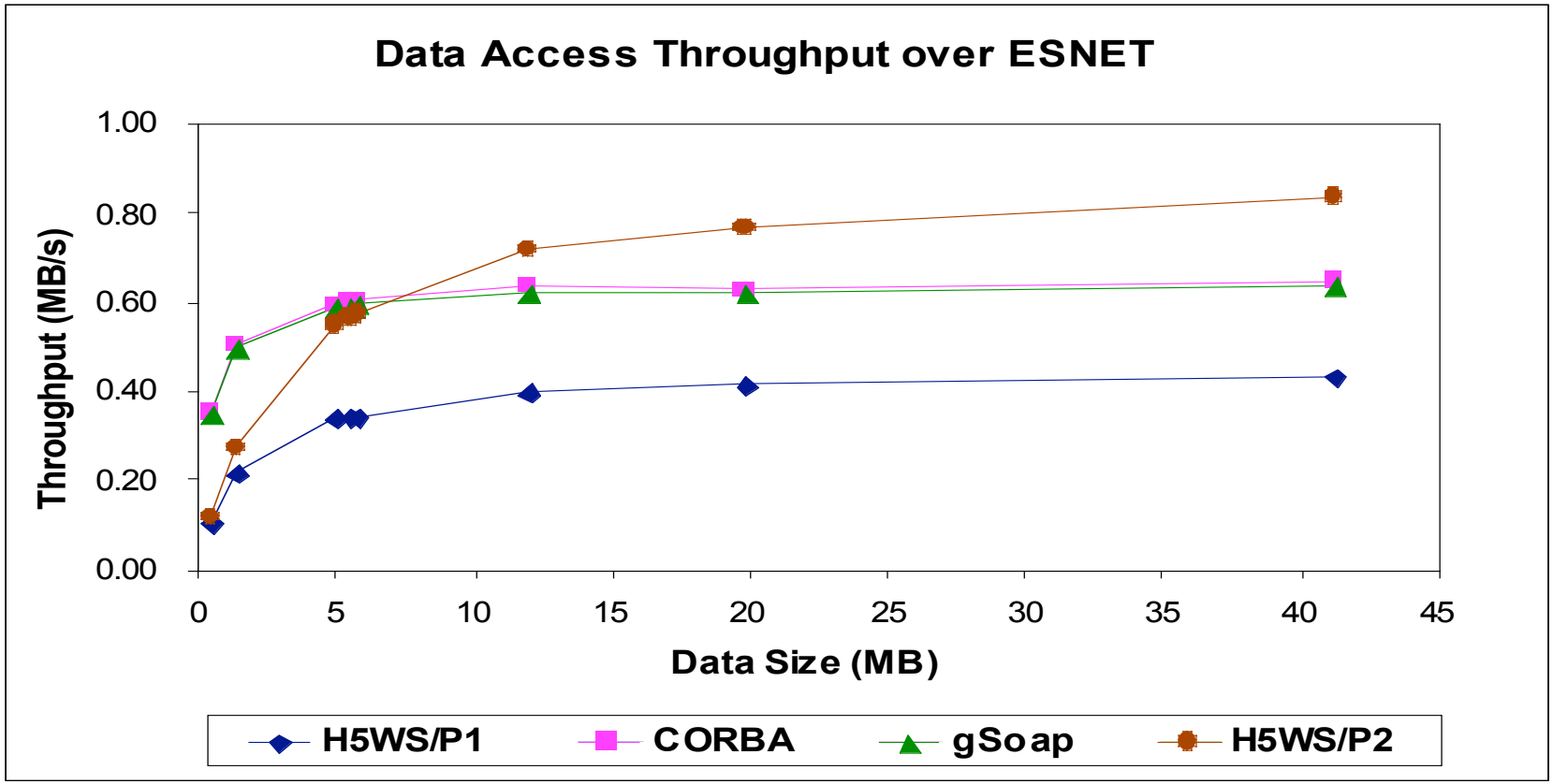
RTT - round trip time

BDP - bandwidth delay product defines how much data is in the pipe

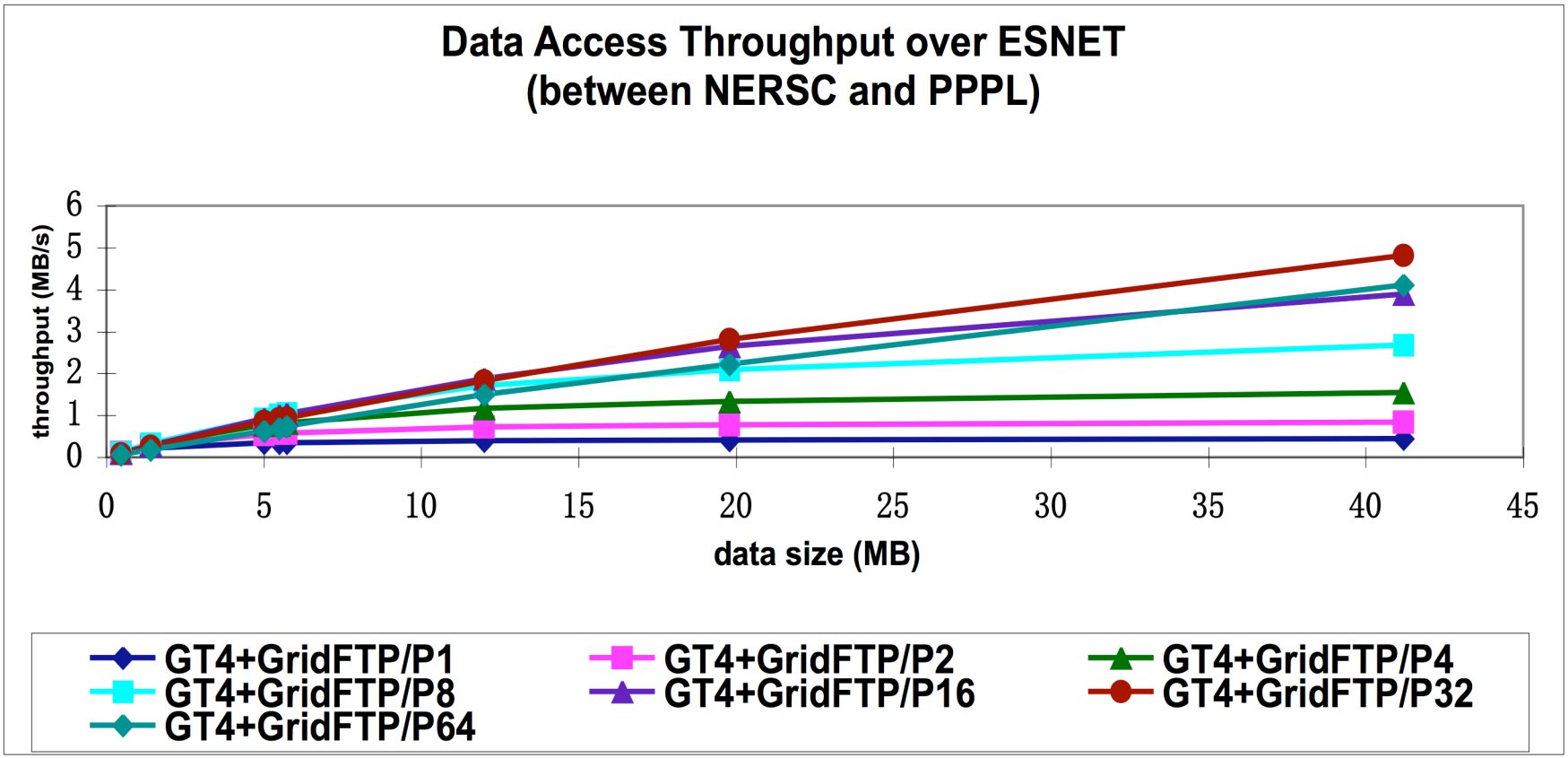
In LAN, H5WS is too expensive (security and temp file), CORBA saturates connection, gSoap is in between



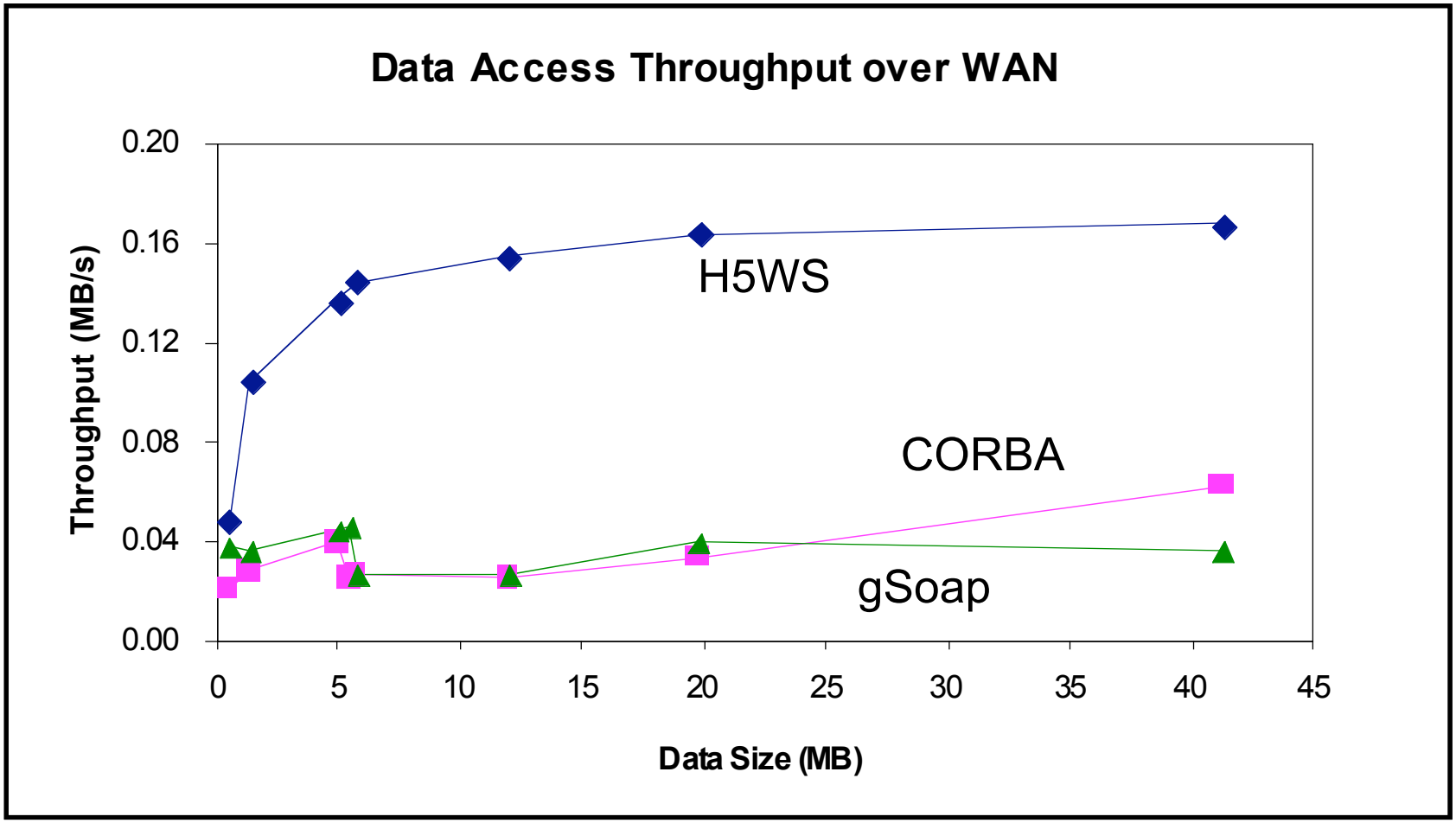
In ESNet H5WS wins with 2 streams, all technologies do not reach the bottleneck (need larger buffers)



Many streams of H5WS work well for wide long pipe (ESNet)



H5WS wins in WAN and is close to bottleneck, CORBA and gSoap underperform



Conclusions-1

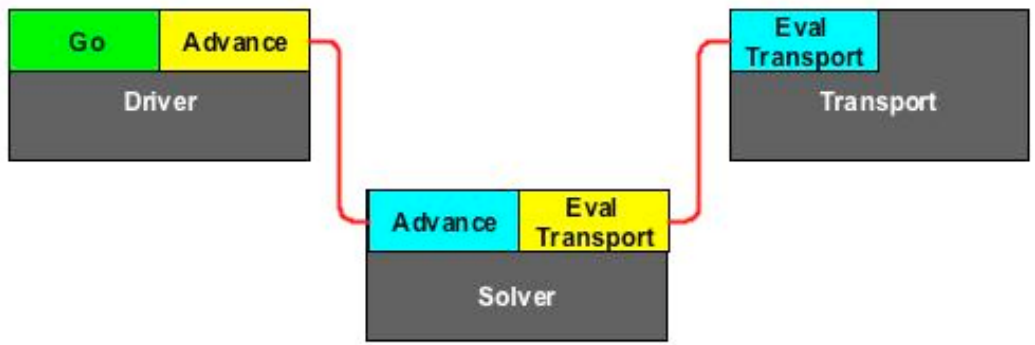
- **Using Globus C WS with GridFTP gave the best overall performance (LAN is not important) but**
  - Imposition of their build system and directory structure
  - Hacking to create Makefiles to add INCLUDES and LIBS
  - Putting implementation into the generated directory (or one can use dlopen with implementations in shared libs). What should be in the repo then?
  - Had to ask for a patch that would allow use of service by people who did not deployed it
  - Debugging nightmare (you compile first, then log in as *globus* user and deploy; might deploy or not, might not run - start over as you)
  - Running all services in one container
- **You might find other solutions that are:**
  - Lighter (gSoap, gLite, ?)
  - Easier to develop (?)
  - Have better support for scientific data types than WSDL (CORBA and CCA/Babel/RMI)

Components: compose, reuse and interchange parts

- **Interact through interfaces (encapsulation)**
- **Separate implementations from interfaces (can swap implementation)**
- **Can be deployed independently**
- **Distinguish between two kinds of interfaces**
  - Provides-port (part of the interface that describes what is offered and can be used by other components)
  - Uses-port (part of the interface describing what is needed)
  - Provides and Uses ports connect to compose an executable
  - This requirement separates components from objects (which can support first two requirements through good design, abstract classes and inheritance, can be deployed as libraries, use pointers to connect)

Non-linear diffusion equation can be solved using three components

$$\frac{\partial \phi(x,t)}{\partial t} = \frac{\partial}{\partial x}\left(D(\varphi, x) \frac{\partial \phi}{\partial x} \right)$$



- **Driver** holds time and through cycles (starts by being called by the framework to *Go*). It asks **Solver** to *Advance*.
- **Solver** knows how to *Advance* the field (given time step and diffusion coefficient) but needs **Transport** to *EvalTransort* (calculate D)
- **Transport** model knows how to *EvalTransort*

Using three components makes sense if you need to solve many diffusion equations

- **One can have many components with the same ports but different implementation**
  - Work with multiple solvers (different discretization schemes, different implicitness)
  - Use multiple diffusion models (glf23, mmm95 for anomalous transport in tokamaks, for example)
- **Swap components without disturbing all of them as the interface and obtain many combinations**

Scientific applications have more requirements

- Support for multiple languages (integrated modeling requires combining multiple legacy codes in one executable). Fortran, specifically!
- Support for High Performance Computing (everything is petascale nowadays)
- Remote invocation (separation of remote HPC part from local data analysis and visualization)

Commercial components do not satisfy additional criteria

- **CORBA supports most of the requirements but**
  - No HPC
  - No multilanguage implementations of CCM (CORBA Component Model) - only Java and C++.
  - No Fortran even for regular CORBA objects.
- **Java Beans**
  - language specific
  - no HPC
- **COM**
  - is platform specific (Microsoft)
- **Plus natural mistrust of scientists to commercial products**

Scientific community started their own frameworks

- **ESMF (Earth Systems Modeling Framework)**
  - Domain specific
  - C++
  - No two types of interfaces
- **SWMF (Space Weather Modeling Framework)**
  - Domain specific
  - F90
  - No two types of interfaces
- **CCA (Common Component Architecture)**
  - TASCS SciDAC
  - Satisfies all required and desired criteria

CCA is the winner - supports all possible features

	ESMF	SWMF	CCA	CCM
Interface Exclusively	Yes	Yes	Yes	Yes
In and Out Interfaces	No	No	Yes	Yes
Multiple Interfaces	No	No	Yes	Yes
Events	No	No	~Yes	Yes
Asynchronous Calls	No	No	Yes	Yes
Attributes	No	No	No	Yes
Internal State and Multiple Instances	Yes	No	No	Yes
Language Neutrality	No	No	Yes	Yes
Separation of Implementation	Yes	Yes	Yes	Yes
Distributed Components	No	No	Yes	Yes
High-Performance Support	Yes	Yes	Yes	No

Babel tool can be useful in mixed language applications

- Part of CCA
- Bindings for C, C++, Java, Python and Fortran
- Great support from Babel developers
- Uses Scientific Interface Definition Language (SIDL) for describing interfaces
- SIDL is valuable for language independent description of interfaces even if you do not generate binding
- SIDL is similar to C++ but adds *in*, *inout*, *out* qualifiers for arguments (similar to CORBA IDL) :

```
package newPackage version 1.0{
  class newClass{
   void doWork(inout double varX );
  }
};
```

Babel is a key tool for multilanguage interoperability and location transparency

- **Implementation in F compiled as a library:**

```
recursive subroutine newPackage_newClass_doWork_mi
       (self, varX, exception)
              !Implementation goes here
end subroutine newPackage_newClass_doWork_mi
```

- **Called from C++ as:**

```
main(){
   newPackage::newClass B = newPackage::newClass::_create( );
   B.doWork(5.); //Done by Fortran
}
```

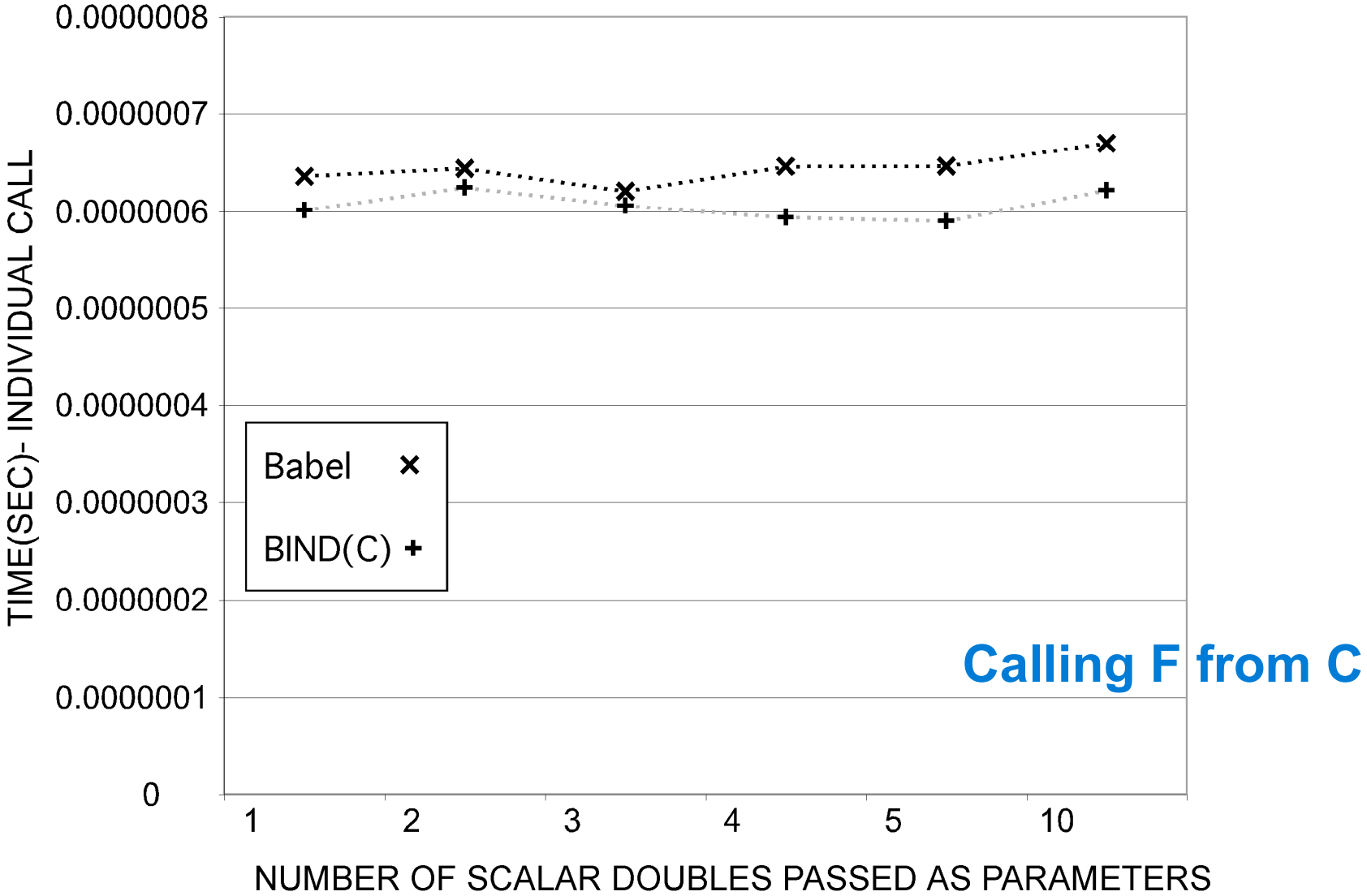
- **Called from Java on a remote machine:**

```
public static void main (String args []) {
   Sidl.rmi.ProtocolFactory.addProtocol
       ("simplehandle", "simple.rmi.SimHandle");
   newPackage.newClass obj = newPackage.newClass.connect
           ("simhandle://hostname:9000/1000");
   obj.doWork (100.0);}
```

Fusion Simulation Project aims for whole device modeling and uses ~components

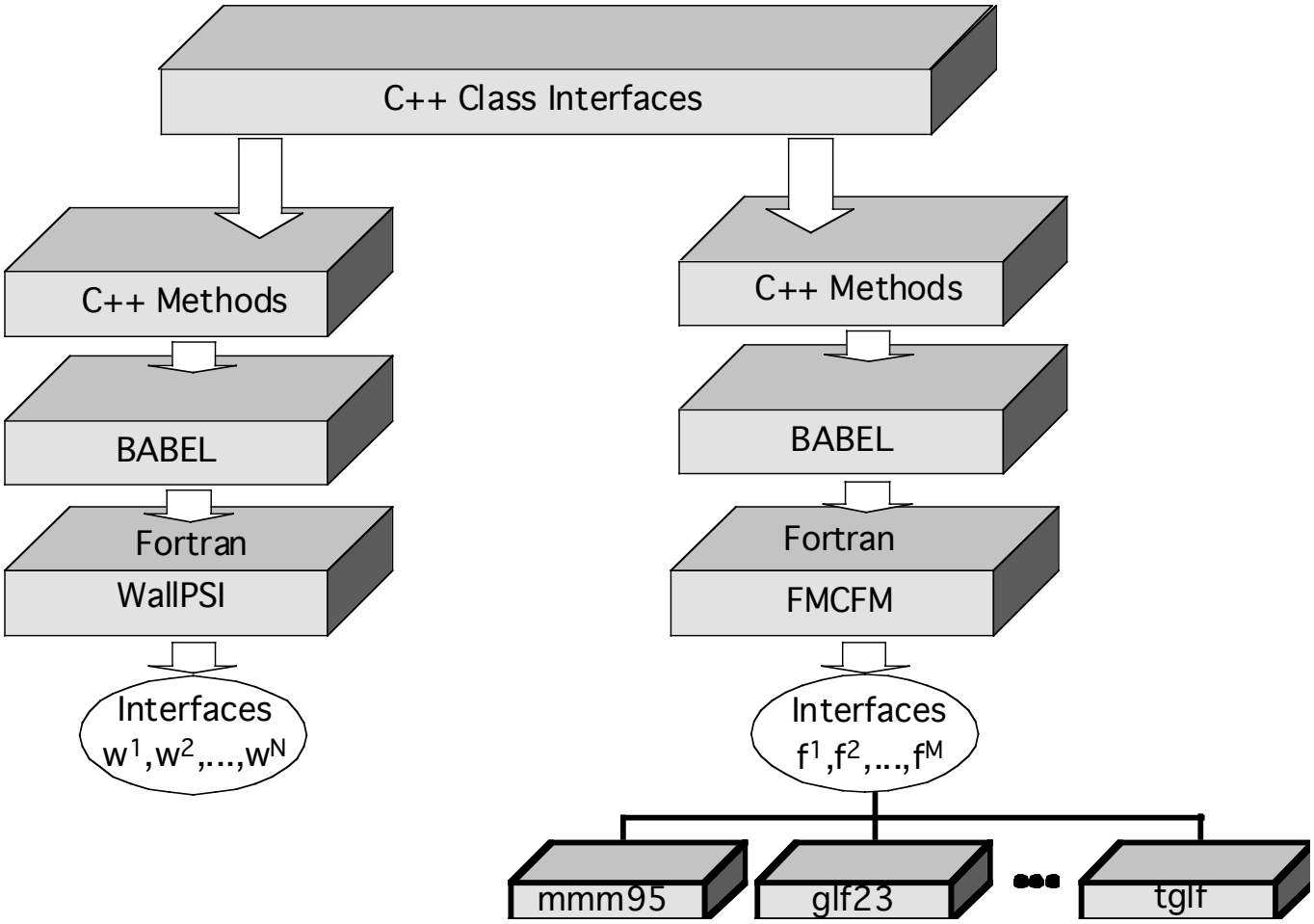
- **Core and Edge - FACETS (Framework Application for Core-Edge Transport Simulations)**
- **MHD and RF - SWIM (Simulation of rf Wave Interaction with Magnetohydrodynamics)**
- **Edge - CPES (Center for Plasma Edge Simulations)**
- **Look at components but use very loose definition of those**
  - SWIM has Python wrappers for codes
  - FACETS has C++ wrappers and uses Babel to call out to Fortran transport, edge and wall codes
  - CPES uses Kepler (loosely coupled Java objects) to describe workflows
- **Europeans (ITM - ITER Tokamak Moleling):**
  - XML for interfaces
  - Kepler for orchestration

FACETS tested Babel before committing and confirmed good performance



Calling F from C

Transport modules are wrapped into SIDL in FACETS



Conclusions-2

- **Many ways to define components, many levels of commitment to component model**
- **CCA is the most promising framework**
- **Migration to CCA is not easy**
  - Define and standardize interfaces using the codes' native languages
  - Express interfaces in SIDL
  - Can now use in multilanguage applications and might stop here, or
  - Define Uses and Provides ports using SIDL
  - Compose using CCA tools
  - Ready to go!
- **Migration to CCA will be easier soon using Bocca (no need to know SIDL in details)**
- **For workflows you might want to look at Kepler.**