



# Distributed Components for Integrating Large-Scale High Performance Computing Applications

Nanbor Wang and Johan Carlsson  
{nanbor, johan}@txcorp.com  
Tech-X Corporation  
Boulder, CO

Compframe Workshop, June 23, 2005

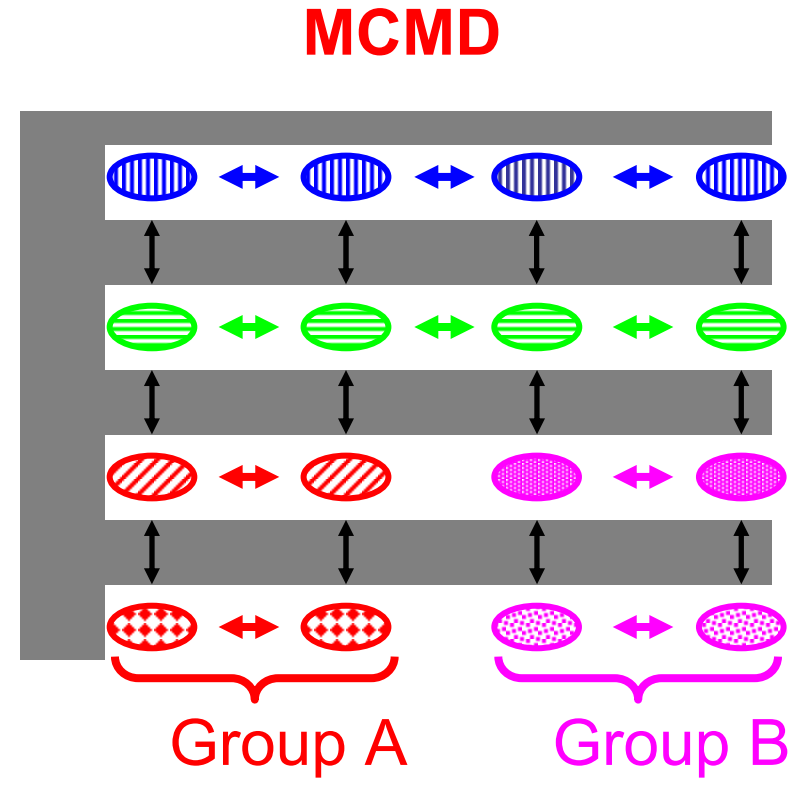
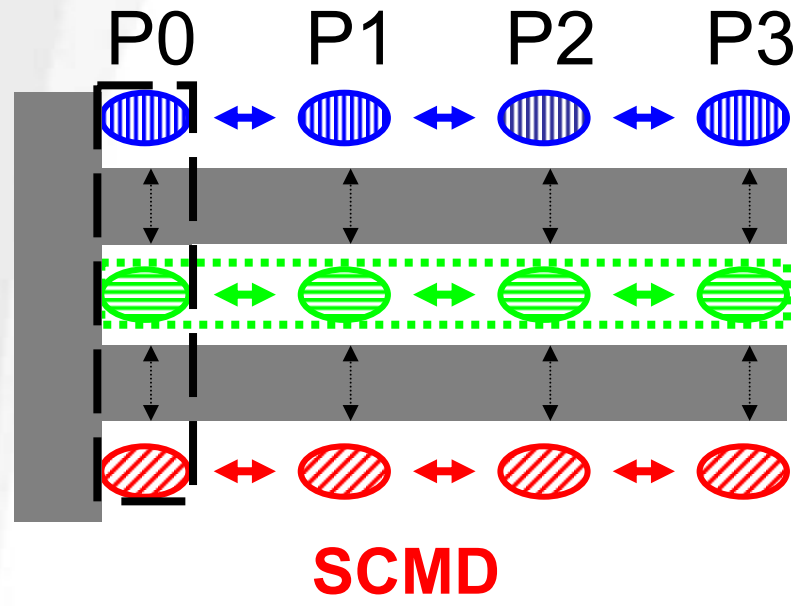
Funded by DOE OASCR SBIR Grant #DE-FG02-04ER84099



# Outlines

- What is Distributed and Parallel High-Performance Computing (DPHPC)
- Motivations
- Distributed components
  - Implementations
  - Current progress
- Future Work

# Existing Parallel CCA Frameworks

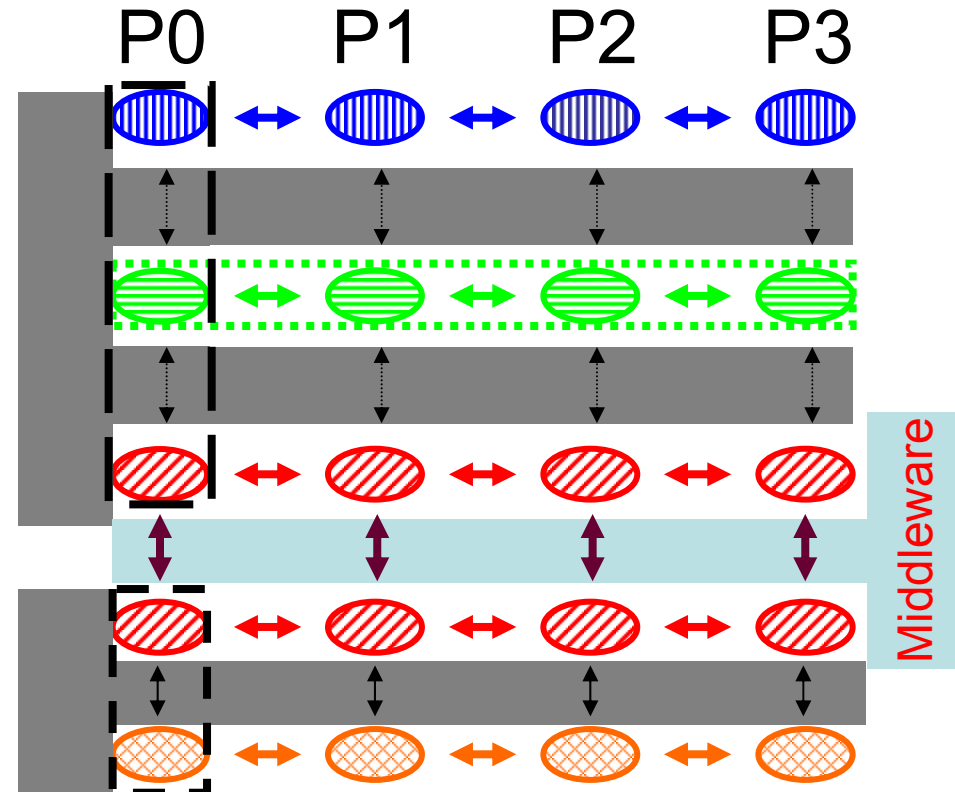


- Support both SPMD and MPMD scenarios
- Stay out of the way of component parallelism
  - Components handle parallel communication



# An Illustration of DPHPC Application

- Still support conventional CCA component managed parallelism
- Provide additional framework mediated distributed inter-component communication capability





# Motivations for Mixing Distributed Tech. and Parallelism

- Provide another high-level abstraction for HPC infrastructure
  - A new dimension for partitioning application compositions
- Motivating example scenarios:
  - Integrate separately-developed and established codes – FSP
  - Provide a different paradigm for partitioning problems – multi-physics simulations
  - Provide ways to better utilize high-CPU number hardware
  - Combine computing resources of multiple clusters/computing centers
  - Enable parallel data streaming between computing task and post-processing task
- What we need: A Distributed and Parallel High-Performance Computing (DPHPC) Environment



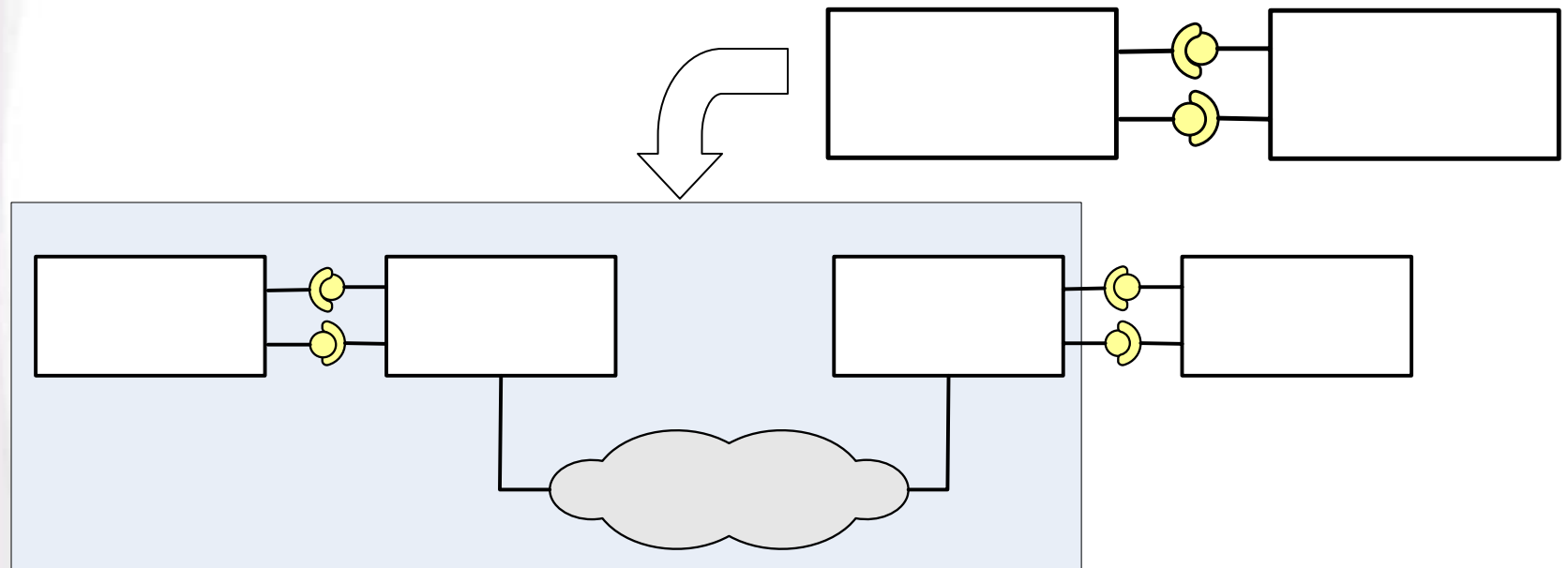
# Solution Approach

- Goal – explore the challenges of developing DPHPC applications
- Existing CCA implementations don't support both models
  - Demo on integration done before but not part of distributions
- Existing parallel codes can be converted to CCA components
  - Using CCA implementations tailored for parallel programming
- Approach
  - Connect distributed parallel CCA applications using well-accepted tools



# Remoting CCA Components

- Connect distributed parallel computations by composing remote-capable proxy components into applications
- Hide the distributed aspect from the localized parallel CCA framework
- Provide low-cost mechanisms for connecting incompatible CCA infrastructures, e.g., Ccafeine, Dune, Ccain, and SciRUN





# Preliminary Work

- Implement prototype distributed components
- Benchmark the performance of distributed components
- Technical details
  - (Almost) current CCA-tools (0.5.6)
  - Two distributed middleware technologies
    - CORBA (TAO)
    - Web/Grid Services (gSoap)
  - Simple strategy for connection configuration
    - Predefined connection endpoints



# Benchmark Operations

- Measuring the throughput for invoking a simple `cube_double` operation

```
interface Cubit extends gov.cca.Port
{
    double cube_double (in double x);
}
```

- CORBA interface

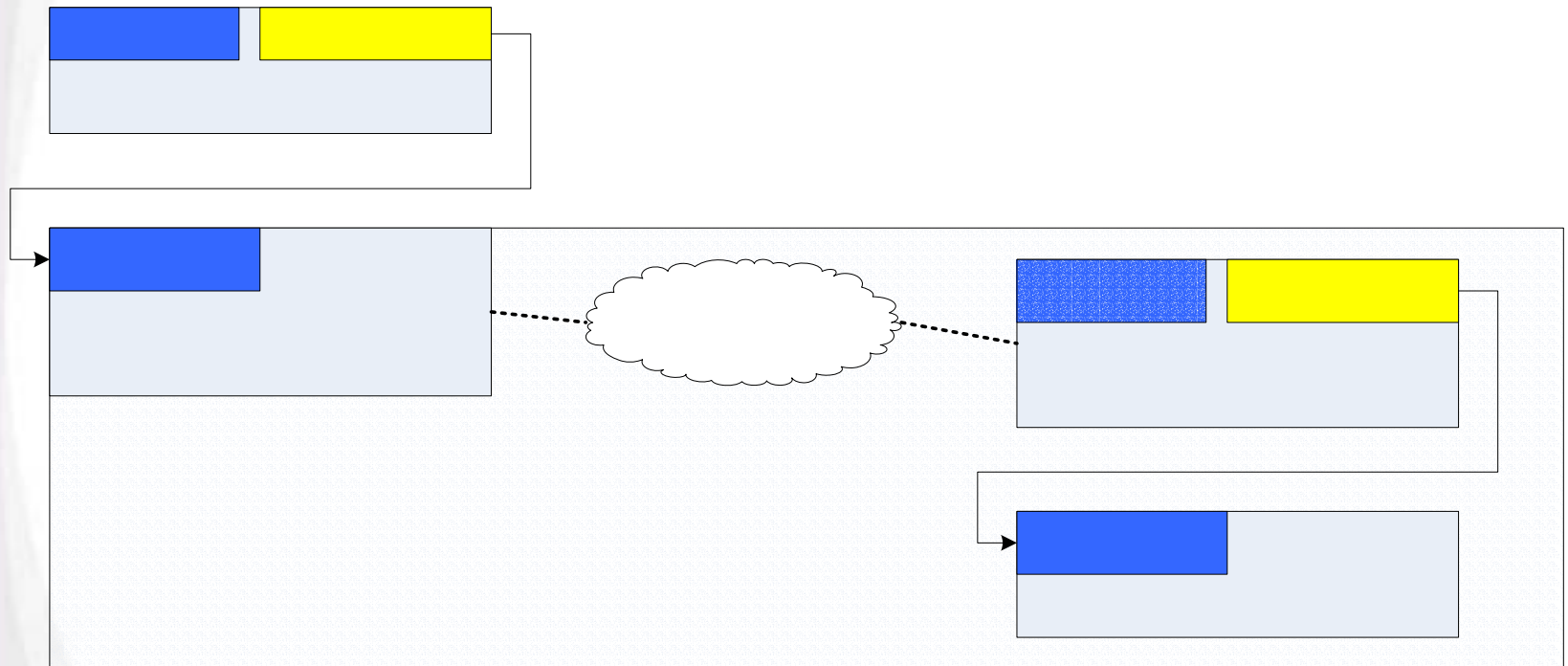
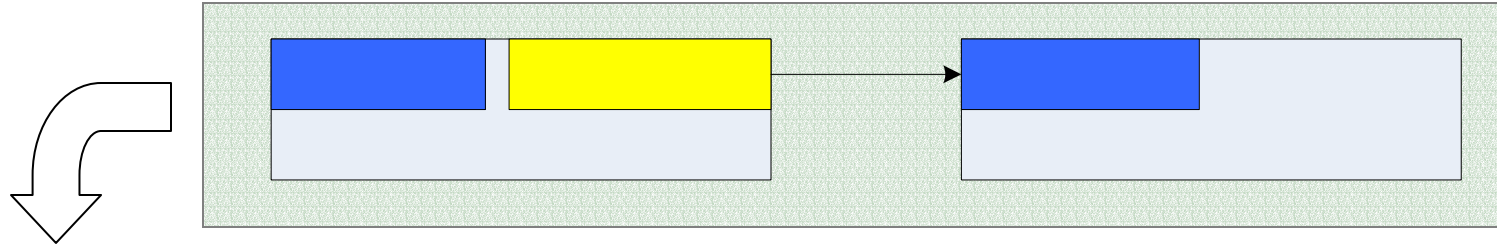
```
interface Cubit
{
    double cube_double (in double x);
}
```

- gSOAP interface

```
int gbenchmark__cube_double (double x, double &r);
```



# Benchmarking Configuration





# Performance Results

- Baseline results (sanity check)
  - Measuring call between two local components with C++ calls

	Direct function calls	Virtual function calls	Component interface calls
Throughput (Mcalls/sec)	103	82.6	36.5

- Cost for remoting operations far exceeds local CCA component calls
- Distributed applications with remoting components can perform equally good



# Performance Results (II)

- Making the application distributed by composing remoting components of different distributed technologies

Interface Type (Calls/sec)	CORBA (TAO)	Web Services (gSoap)
Client/Server (local loopback)	8674	6162
DistComp (local loopback)	8386	6030
DistComp (LAN)	2359	1384

- Distributed components add minimal overhead
- CORBA provides better throughput even with short messages/frequent interactions
- Results from other measurements show CORBA outperforms gSoap with large datasets
- Note: Ongoing development of binary XML



# Future Works

- Overall – aspects in providing and using a DPHPC environment
- Planned tasks:
  - Hardening of distributed component implementations and tools
  - Support user-defined data structure available in modern languages
  - Examine and review different composition strategies for DPHPC applications
  - Develop example DPHPC applications



# Hardening of Distributed Component Implementations

- Proxy-style remoting components
  - Examine carefully the mapping from SIDL to CORBA IDL
  - Develop prototype tools for generating remoting component implementations
- Integration with Babel RMI APIs
  - Implement CORBA-based Babel RMI library
  - Implement other efficient IPC mechanisms for certain hardware configurations
  - Explore Proteus



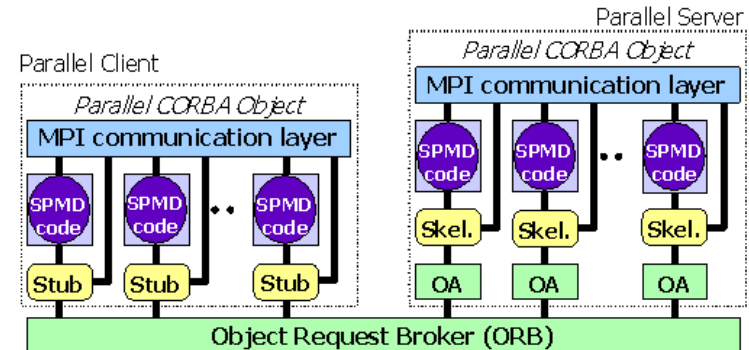
# Support Modern Language Constructs for DPHPC

- Discussion with LLNL researchers: struct is a most requested features
- Work on FSP proposal also calls for “struct” support
- Need to support for both local and distributed cases
- Will collaborate with LLNL researchers
  - BNF and AST extension
  - Code generation for C/C++, FORTRAN
  - Relationship with FORTRAN Bind (C)



# Examine Deployment Strategies for DPHPC Applications

- Local-CCA component centric view:
  - Local applications
  - Employ a distributed “builder service” for registering/requesting distributed ports
- Distributed component centric view:
  - Two-tier deployment – remote components and their implementations
- Grid view:
  - Making distributed components as grid services
- Synergy with distributed CCA implementations



Research on parallel components:  
 PACO++ by Paris Group in IRISA,  
 France  
 (<http://www.irisa.fr/paris/General/>)



# Develop Example DPHPC Applications

- Running HPC applications using multiple clusters
  - Utilize ORNL LDRD fusion CCA components
- Running HPC applications using large-cpu-count hardware
  - Collaborate with the FSP team
- Connecting HPC applications with online data analysis in real-time
  - Utilize parallel data streaming



# Concluding Remarks

- Our goals are:
  - Provide an environment for DPHPC applications
  - Document usage patterns of developing DPHPC applications
- Current status
  - Benchmarking two major transport mechanisms, CORBA and SOAP
  - Two major types of interaction models
    - Frequent short control messages
    - Periodic large datasets
- Future work:
  - Hardening of remoting component implementations and tools
  - Support other modern language construct
  - Examine and review different composition strategies for DPHPC applications
  - Develop example DPHPC applications



# Questions