Moving from FT-CORBA to FT-CCM

MEAD: Middleware for Embedded Adaptive Dependability

Deepti Srivastava, Aaron Paulos & Priya Narasimhan

Carnegie Mellon University
Pittsburgh, PA 15213-3890

16th July 2004

Background

- **MEAD: Real-time fault-tolerant middleware being developed at Carnegie Mellon University**
- **MEAD was born out of the realization that**
  - The Fault-Tolerant CORBA and the Real-time CORBA standards ignore each other completely
  - CORBA applications today can get either real-time support or fault-tolerant support, but not both
- **Objectives of MEAD**
  - Why real-time and fault tolerance do not make a good "marriage"
  - Overcoming these issues to build support for embedded middleware applications that require **both** real-time **and** fault tolerance

MEAD in a Nutshell

- **Resolving trade-offs between real-time and fault tolerance**
  - Ordering of tasks to meet replica consistency and task deadlines
  - Bounding fault detection and recovery times in asynchronous environment
  - Estimating worst-case performance in fault-free, faulty and recovery cases
- **MEAD's RT-FT middleware support**
  - Tolerance to crash, communication and timing faults
  - Proactive fault-tolerance framework
  - Fault-tolerance advisor to take the guesswork out of configuring reliability
  - Offline program analysis to detect, and to compensate for, RT-FT conflicts
- **Primary focus of MEAD was CORBA (TAO)**

Current Release on Emulab – Features

- **Features in MEAD version 1.1**
  - Active replication and warm passive replication
  - Stateful and stateless distributed applications
  - Focus on CORBA applications
  - Tunable parameters: number of replicas, replication style
  - http://www.ece.cmu.edu/~mead/release/index.html
  - Send us email if you are interested in using MEAD
    - mead-support@lists.andrew.cmu.edu
- **Upcoming features in next release**
  - Focus on CCM applications – today's talk
    - Driven by the emerging consideration of CCM for mission-critical applications
  - Tunable parameters: number of replicas, replication style, checkpointing frequency
  - Integrating resource-aware fault-tolerance (i.e., making fault-tolerance decisions based on resource usage information)

Outline of Talk

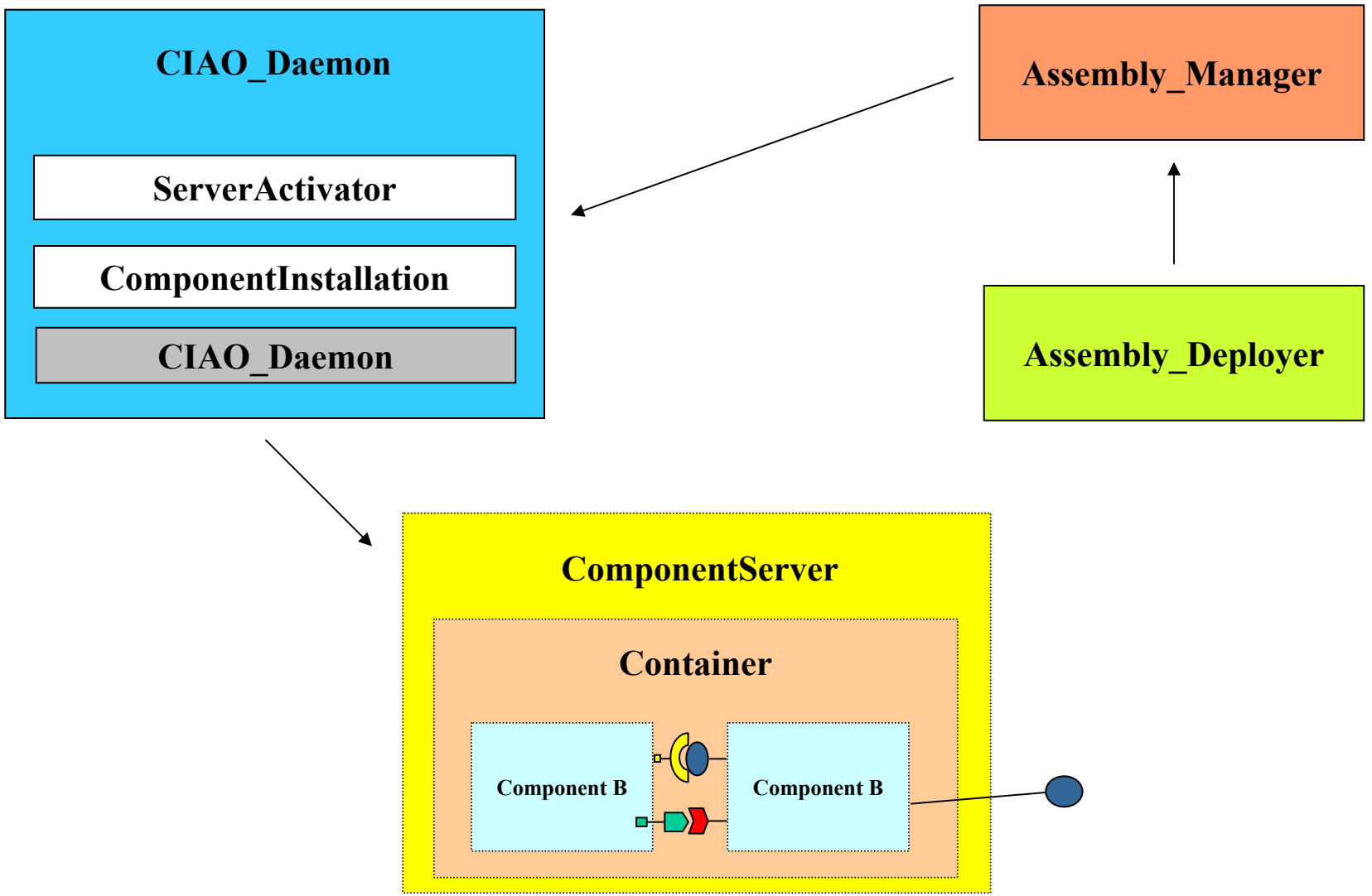
- Motivation
- CCM architecture
- Objectives
- FT – CCM architecture
- Assumptions
- Internal Details
- Preliminary Performance Results
- Challenges in Developing FT-CCM
- Lessons Learnt
- Summary

Motivation

- **Why FT-CCM**
  - CCM technology is currently in early stages of adoption
  - CCM has a potential for large-scale deployment
  - With emerging consideration of CCM technology in mission-critical applications, fault tolerance for CCM will be essential

- **We are uniquely poised to develop a FT-CCM architecture**
  - Leverage domain knowledge of CORBA
  - Fault Tolerance background
  - We are already working on MEAD

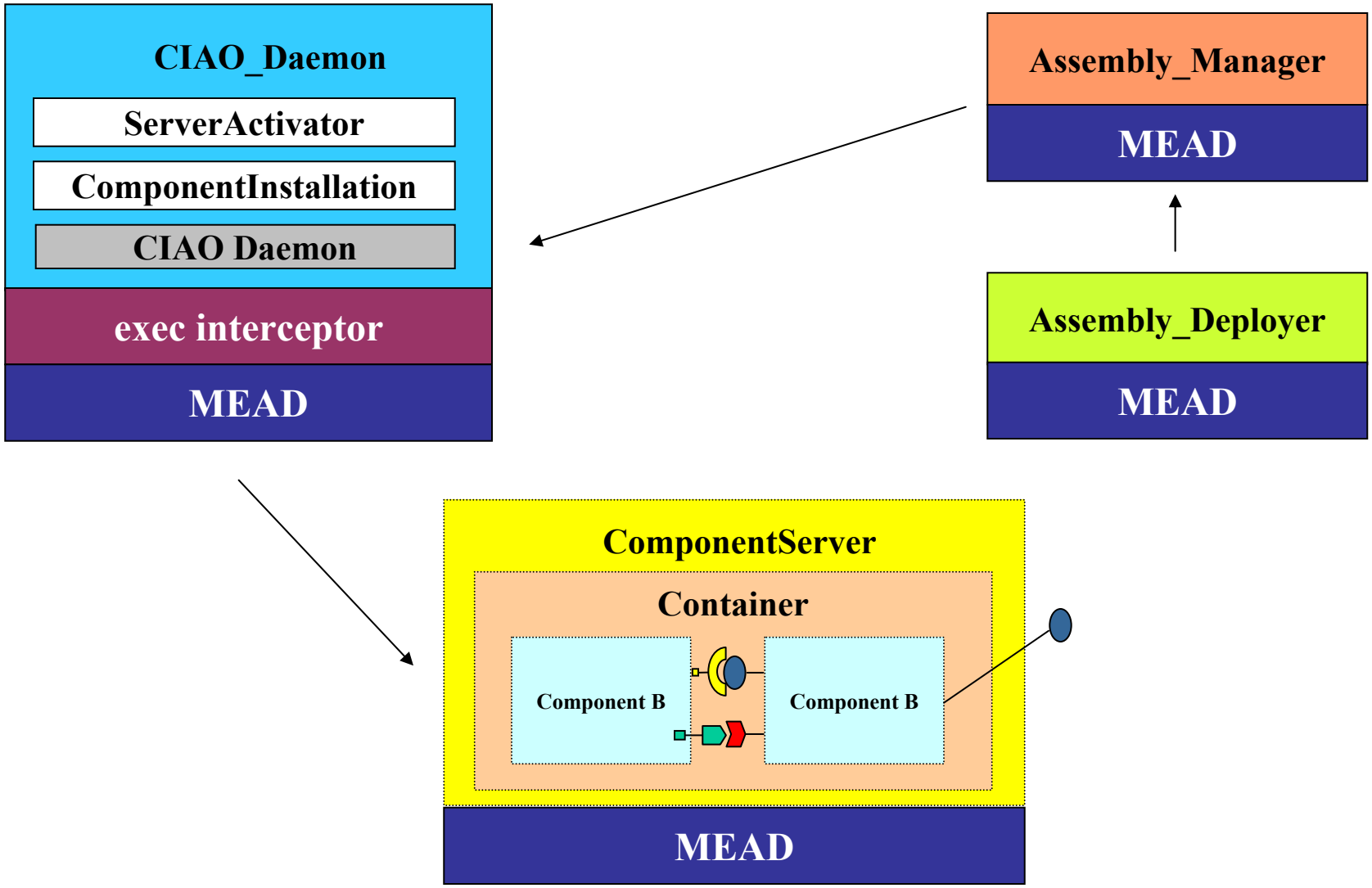
CIAO CCM Architecture



Objectives

- **Investigate and define a Fault Tolerant Model for the CORBA Component Model**
- **Investigate the ease and feasibility of migrating from FT-CORBA to FT-CCM**
  - Identify changes that need to be made to an FT-CORBA infrastructure to add support for a Component model
  - Investigate whether MEAD works out-of-the-box
  - Focus of this talk

FT-CCM Architecture



Current Working Assumptions

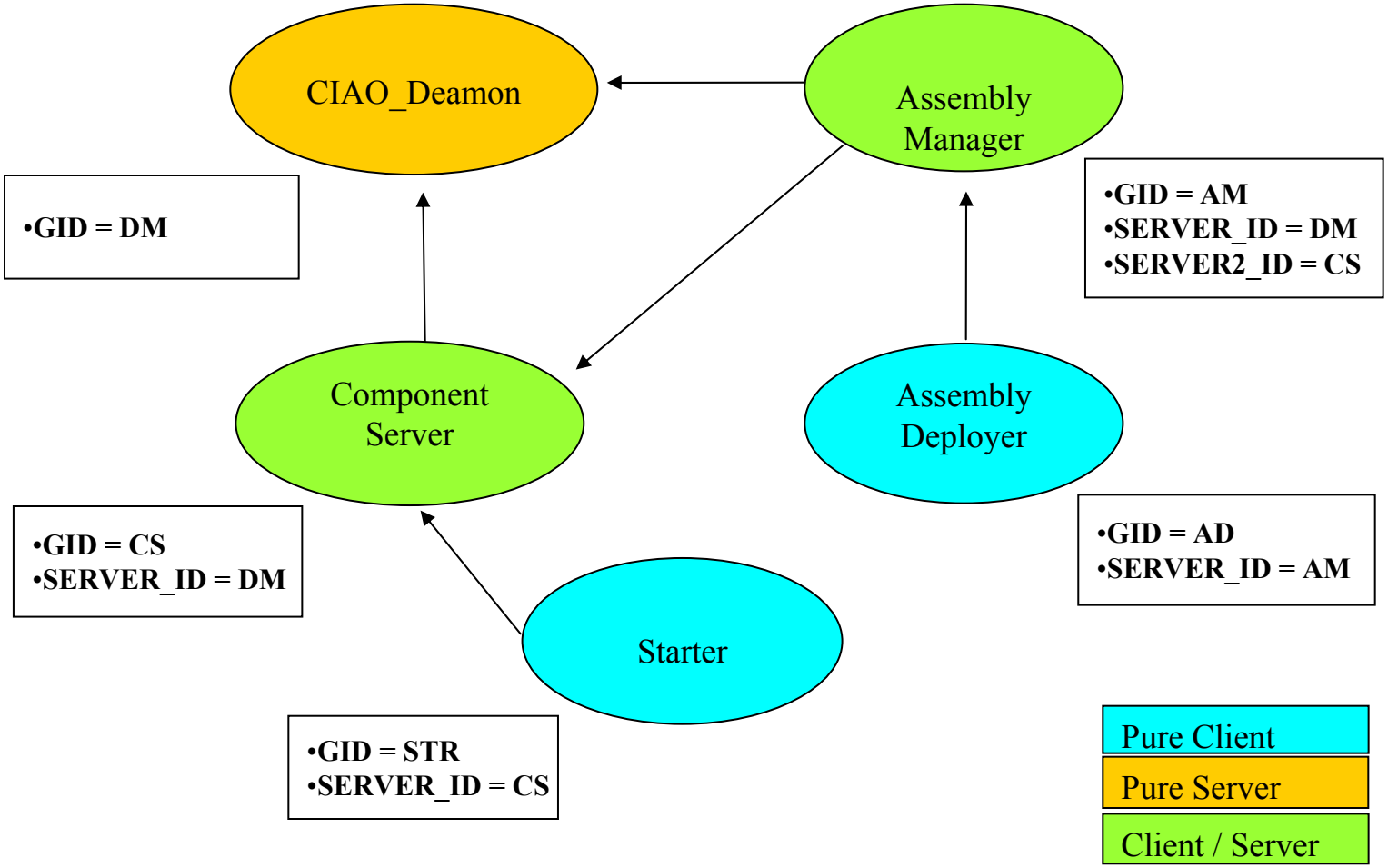
- Only replicating the Component Server
- Not replicating the CIAO deployment infrastructure including:
  - Assembly Manager
  - Assembly Deployer
- Also ignore that these are single points of failure in the CCM architecture
- Assume no state in the Component Server
- Assume the Components are stateless

Internal Details

- **Environment Setup for MEAD + Spread**
  - Setup the connections so all communication is via MEAD (Spread)
  - Identify the roles of clients and servers in the CIAO deployment infrastructure
  - No way to specify execution environment for the Component Server
- **The "exec" interceptor**
  - Dynamically loaded library which interposes the fork and exec calls
    - Sets up the environment to launch process with MEAD
    - Launches component server with MEAD
    - Launches CIAO_Daemon (or Daemon Controller) with MEAD

Internal Details

$CIAO_ROOT/examples/Hello Communication



Internal Details

- **Object Persistence**
  - Replication requires CORBA object keys to be persistent
  - The object keys created by default are transient
  - Create POA policy for persistent lifespan

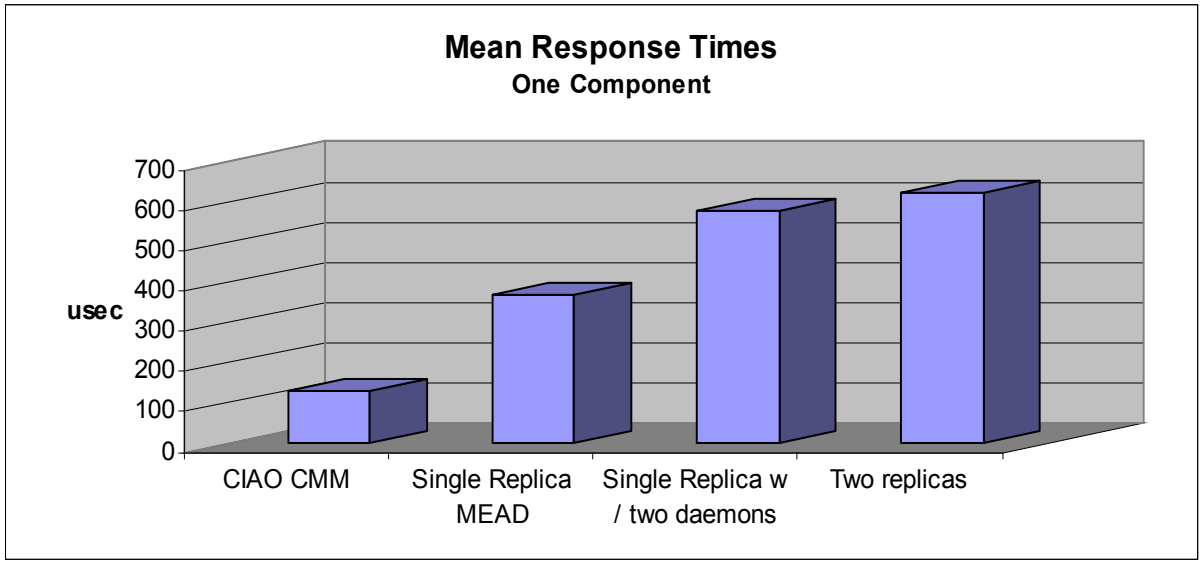
- **Multiple connections to the same process**
  - Component Server houses container and components
  - Support at client for multiple connections to the same component server process
    - Separate connections to container (for creation/destruction of component) and to component (for invocations)
    - Maintain internal mapping in MEAD of multiple FDs to same spread connection

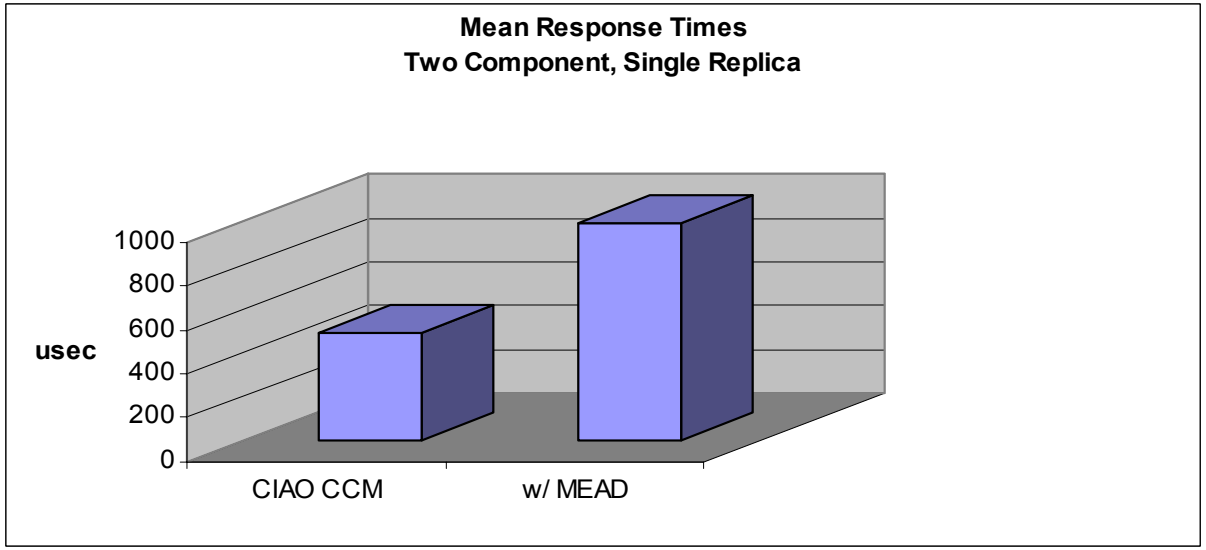
Experimental Setup

- **Using CIAO implementation of the CCM specification**
  - Version – 0.4.1
  - $CIAO_ROOT/docs/tutorial/Hello example – One component
  - $CIAO_ROOT/examples/Hello – Two components

- **Testbed**
  - Hardware – Intel Pentium 4, 2.4 Ghz with 512K Cache, 512M, Linux: Kernel 2.4.20
  - Operating System – Redhat 9
  - 100 Mbps Ethernet
  - MEAD version 1.1
  - Spread version 3.17.1

Preliminary Performance Results





Challenges in Developing FT-CCM

- **Understanding the process launch mechanisms in ACE+TAO+CIAO**
- **Understanding the internal details of the CIAO Implementation required to deploy with MEAD**
  - Interactions between objects during deployment and installation
  - Interaction between objects during client invocations
- **Support for IIOP callbacks**
- **POA Persistence**
  - Locating and understanding the usage of POAs in CIAO source
  - Identifying activation of relevant objects using the POAs

Challenges – Looking Forward

- **Support for object communication in environments that use multiple objects in a process**
  - Multiple objects located in the Component Server inherent the same MEAD GID. This makes it difficult to distinguish between reply messages from these objects at the MEAD level.
- **Replicating the deployment infrastructure and the CIAO daemon**
  - These are single points of failure
  - Implications
- **Investigating if Component Servers maintain state**
- **Support for replication in stateful CCM applications**

Lessons Learned

- **MEAD does work out-of-the-box**
  - Modifications to MEAD
    - Support for multiple connections to the same process
- **Standard IIOP is supported in both models**
- **Steep learning curve**

Summary

- Overview of MEAD
- Overview of CCM and Proposed FT-CCM architecture
- What it takes to migrate from FT-CORBA to FT-CCM
- Challenges
- Lessons learnt

For More Information on MEAD

Electrical & Computer ENGINEERING

http://www.ece.cmu.edu/~mead