Technology for Performance Engineered Systems

From the 9/19-20/96 DARPA Workshop on Performance:

“

Paradigms:

- Is it possible to design computing systems and their applications, like…we design and build aircraft
  - where there exists a battery of simulation tools, such as CFD models, structural mechanics models, engine combustion models, etc...
  - and also prototypes, wind tunnels, flight simulators

”

The President's Information Technology Advisory Committee

Some findings:

"We cannot safely extend what we currently know to more complex systems"

"Learning how to build large-scale, highly reliable and secure systems requires research"

Recommendation:

"Increase funding in research and development of core software… "

Panel Session at SC'98

Technology for Performance Engineered Computer and Communications Systems

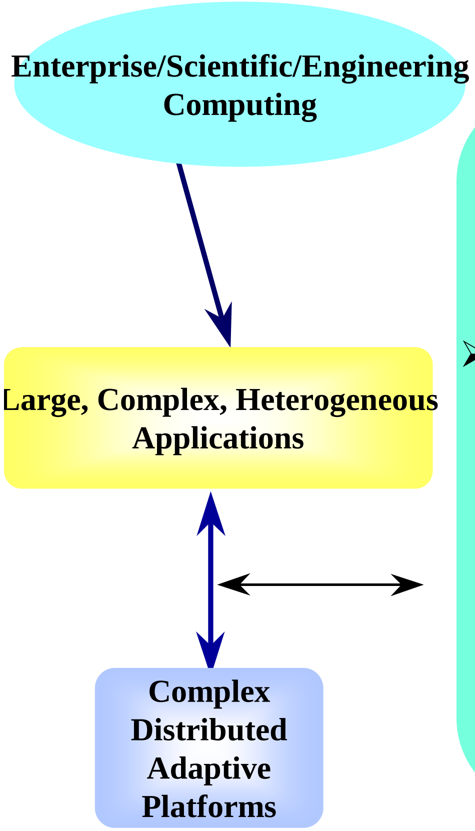
– Friday, Nov 13, 1998, 8:30 - 10:00 AM

Workshop on Performance Technology

- What I’m looking for from this workshop:
  - What’s covered by your projects
  - What other efforts are needed
  - 1-page summaries

Distributed Computing Support (DisCoS)

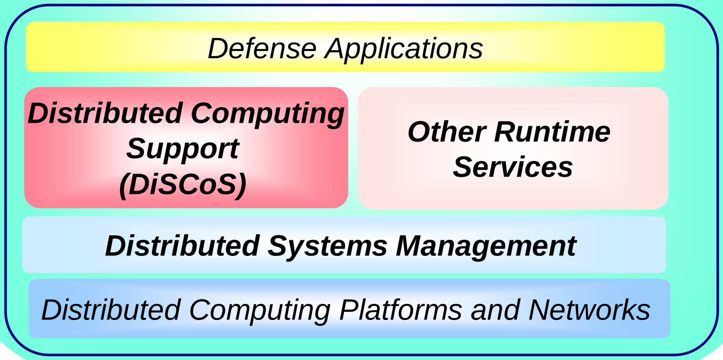
"Empowering Applications to exploit Future Distributed Heterogeneous Computing Systems"


Enterprise/Scientific/Engineering Computing
Large, Complex, Heterogeneous Applications
Complex Distributed Adaptive Platforms

DisCoS

- Distributed compiler and programming model
- Application assembly technology
- Computing system analysis technology

➢ *Enabling applications to efficiently execute across distributed, heterogeneous platforms*


Defense Applications
Distributed Computing Support (DiSCoS)
Other Runtime Services
Distributed Systems Management
Distributed Computing Platforms and Networks

Impact

Management enhanced with distributed applications

Example: Target Recognition

- Distributed execution
- Highly efficient execution
- Real time or faster than real time
- Improve accuracy of analysis

Improved process for application design, support and upgrade

Example: Pattern Recognition

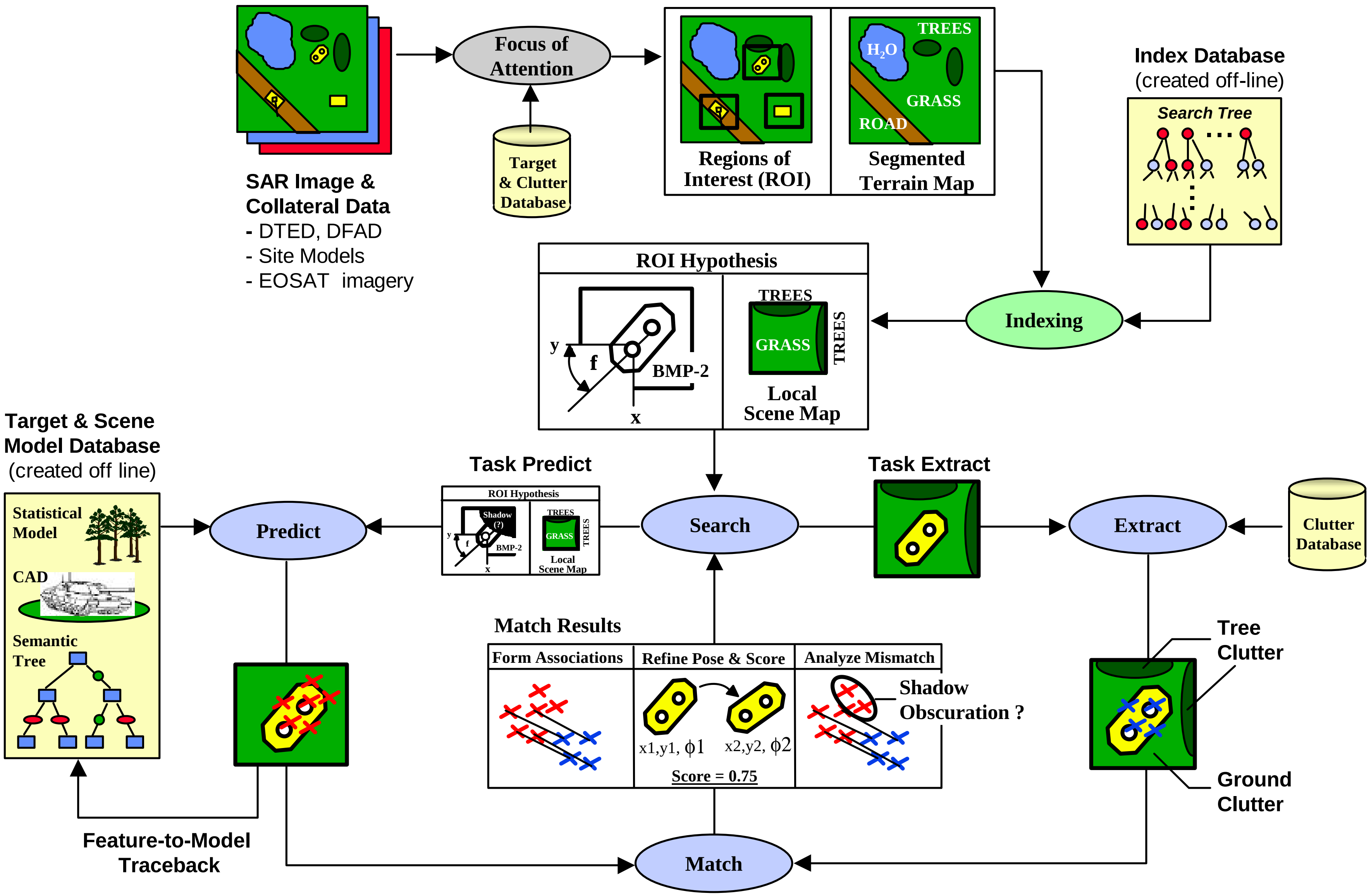
- Reduce time to port applications: from months to ~hour
- Reduce cost to port applications: from $1M/port to $1K
- Enable rapid prototyping
- Reduce prototyping costs

Outline:

- Background: The technology gap
- Case example: DisCoS applied to MSTAR
- New approach
- Why now? Why DARPA?
- Program -PAD

DARPA MSTAR

(Moving and Stationary Target Acquisition and Recognition)



SAR Image & Collateral Data

- DTED, DFAD
- Site Models
- EOSAT imagery

Focus of Attention

Target & Clutter Database

Regions of Interest (ROI)

Segmented Terrain Map

Index Database (created off-line)

Search Tree

Indexing

ROI Hypothesis

Local Scene Map

Target & Scene Model Database (created off line)

Statistical Model

CAD

Semantic Tree

Predict

Task Predict

Search

Task Extract

Extract

Clutter Database

Match Results

Form Associations	Refine Pose & Score	Analyze Mismatch
	x1,y1, ϕ1 x2,y2, ϕ2 Score = 0.75	Shadow Obscuration ?

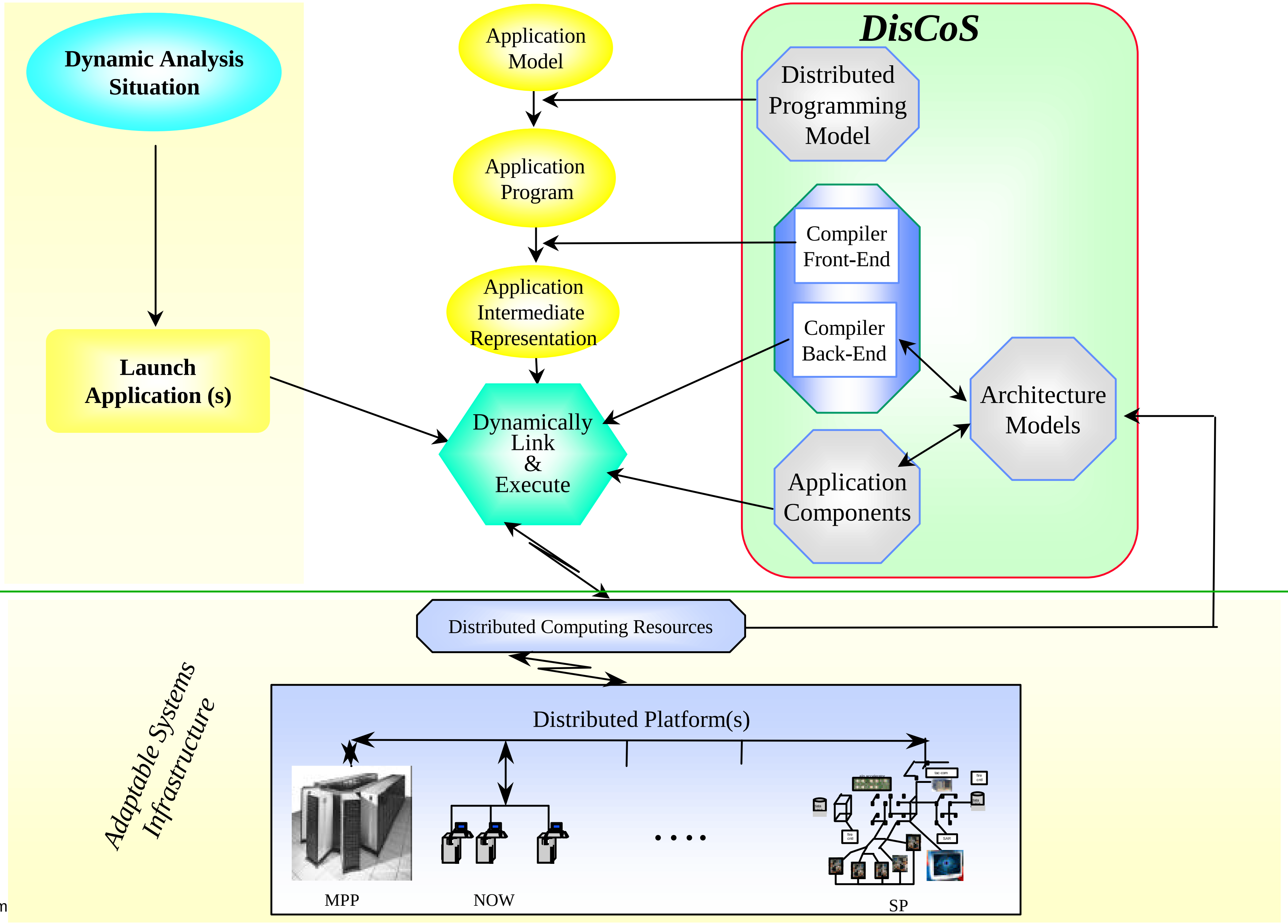
Tree Clutter

Ground Clutter

Feature-to-Model Traceback

Match

The Solution: DisCoS Technology



Dynamic Analysis Situation

Launch Application (s)

Application Model

Application Program

Application Intermediate Representation

Dynamically Link & Execute

DisCoS

Distributed Programming Model

Compiler Front-End

Compiler Back-End

Architecture Models

Application Components

Distributed Computing Resources

Adaptable Systems Infrastructure

Distributed Platform(s)

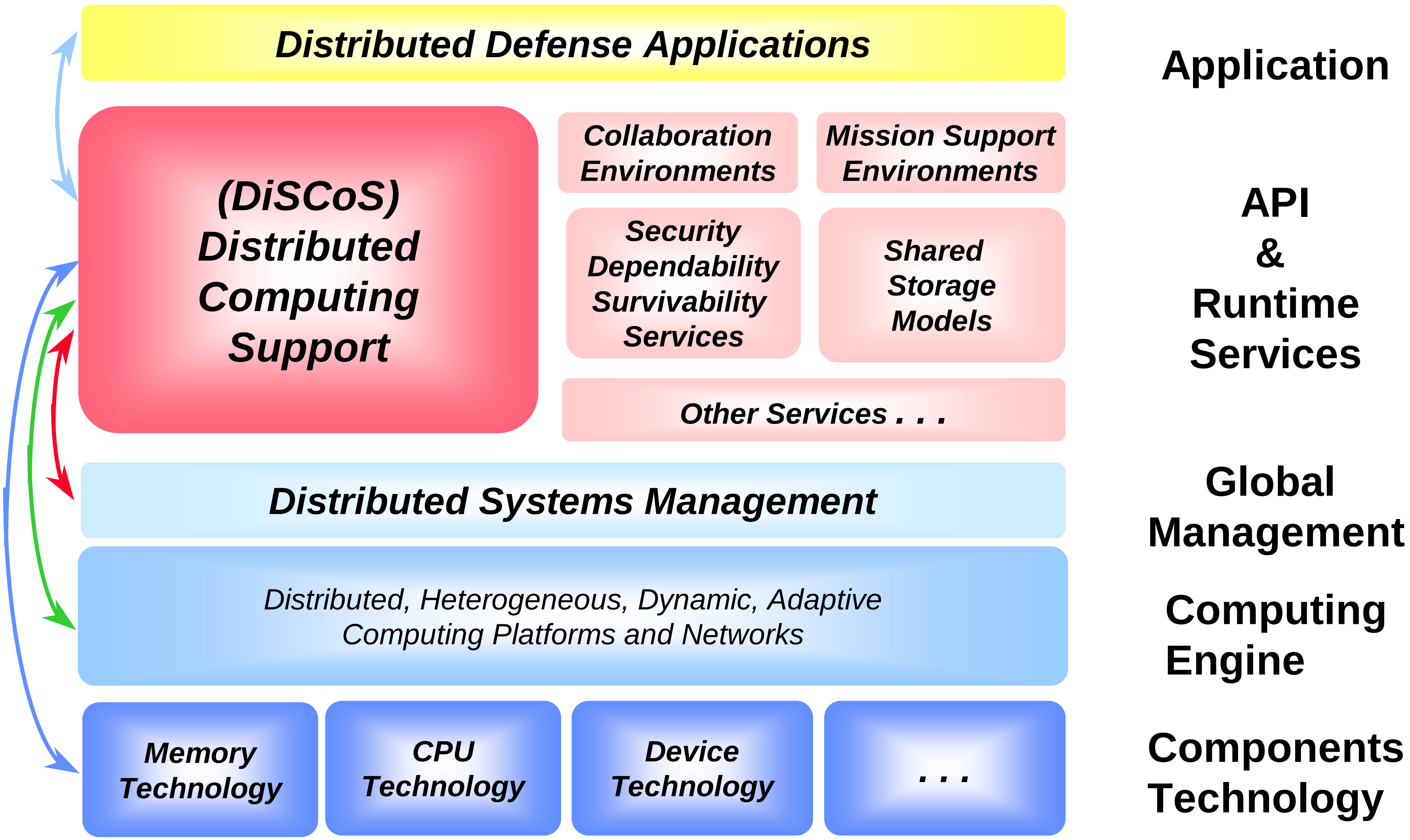
MPP

NOW

. . . .

SP

Distributed Systems Software/Hardware Architectural Framework



Distributed Defense Applications

(DiSCoS) Distributed Computing Support

Collaboration Environments

Mission Support Environments

Security Dependability Survivability Services

Shared Storage Models

Other Services . . .

Distributed Systems Management

Distributed, Heterogeneous, Dynamic, Adaptive Computing Platforms and Networks

Memory Technology

CPU Technology

Device Technology

. . .

Application

API & Runtime Services

Global Management

Computing Engine

Components Technology

Technical Areas

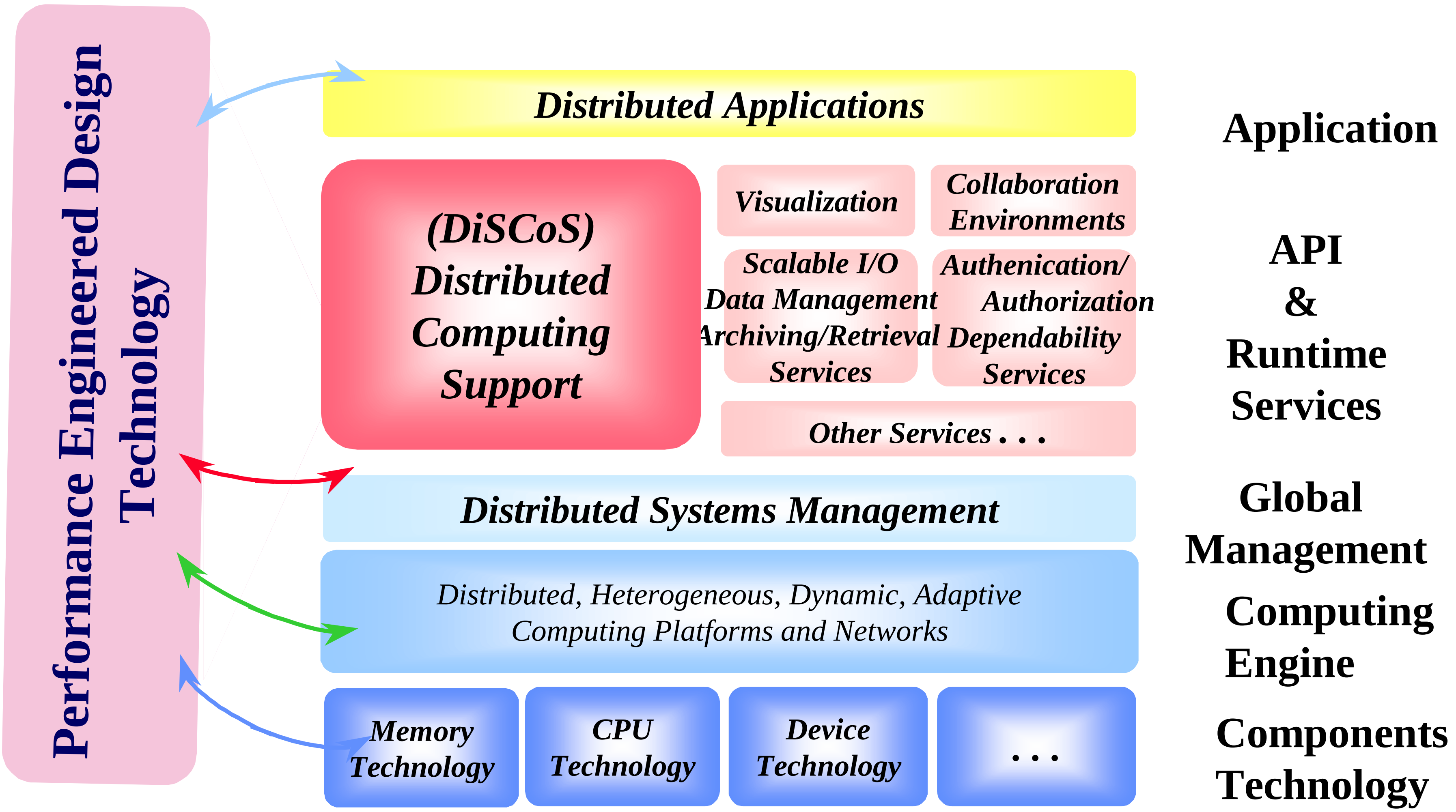
- **Application Programming System** (distributed programming models and compilers)
  - **distributed programming models for complex, distributed hardware platforms with complex memory structure and be adaptable to changes in the underlying platforms**
  - **interfaces that allow applications to specify performance related parameters to enable applications to achieve quality of service**
  - **compilers that interface with models of the underlying distributed hardware and software platforms to allow retargeting and optimizing application mappings on such complex systems**

- **Application Composition System** (dynamic selection of distributed application components)
  - **technology for building knowledge-based systems allowing automatic selection of solution methods allowing applications to adapt to changes in the underlying platforms**
  - **application interfaces and methods for problem specification and extracting content information, standards of interfaces, data representation and data exchange, and standard high-level and low-level libraries**
  - **interfaces to debugging tools and performance models**

- **Application Analysis System** (technology for performance engineered distributed applications)
  - **modeling languages and models for application and system description**
  - **multi-resolution levels of data abstraction for interoperability of performance models of different levels of abstraction**
  - **methods and tools for measurement and instrumentation**

- **Validation, Integration and Demonstrations** (validation, integration and demo of the technology)
  - **validation of key technologies developed under each of the tasks above**
  - **identify integrator to integrate the technologies developed above**
  - **demonstration of the ability of these technologies for design and runtime support of key applications executing under dynamically changing conditions (examples: Target/Pattern Recognition)**

Distributed Systems Software/Hardware Architectural Framework



Performance Engineered Design Technology

Distributed Applications

(DiSCoS) Distributed Computing Support

Visualization

Collaboration Environments

Scalable I/O Data Management Archiving/Retrieval Services

Authenication/ Authorization Dependability Services

Other Services . . .

Distributed Systems Management

Distributed, Heterogeneous, Dynamic, Adaptive Computing Platforms and Networks

Memory Technology

CPU Technology

Device Technology

. . .

Application

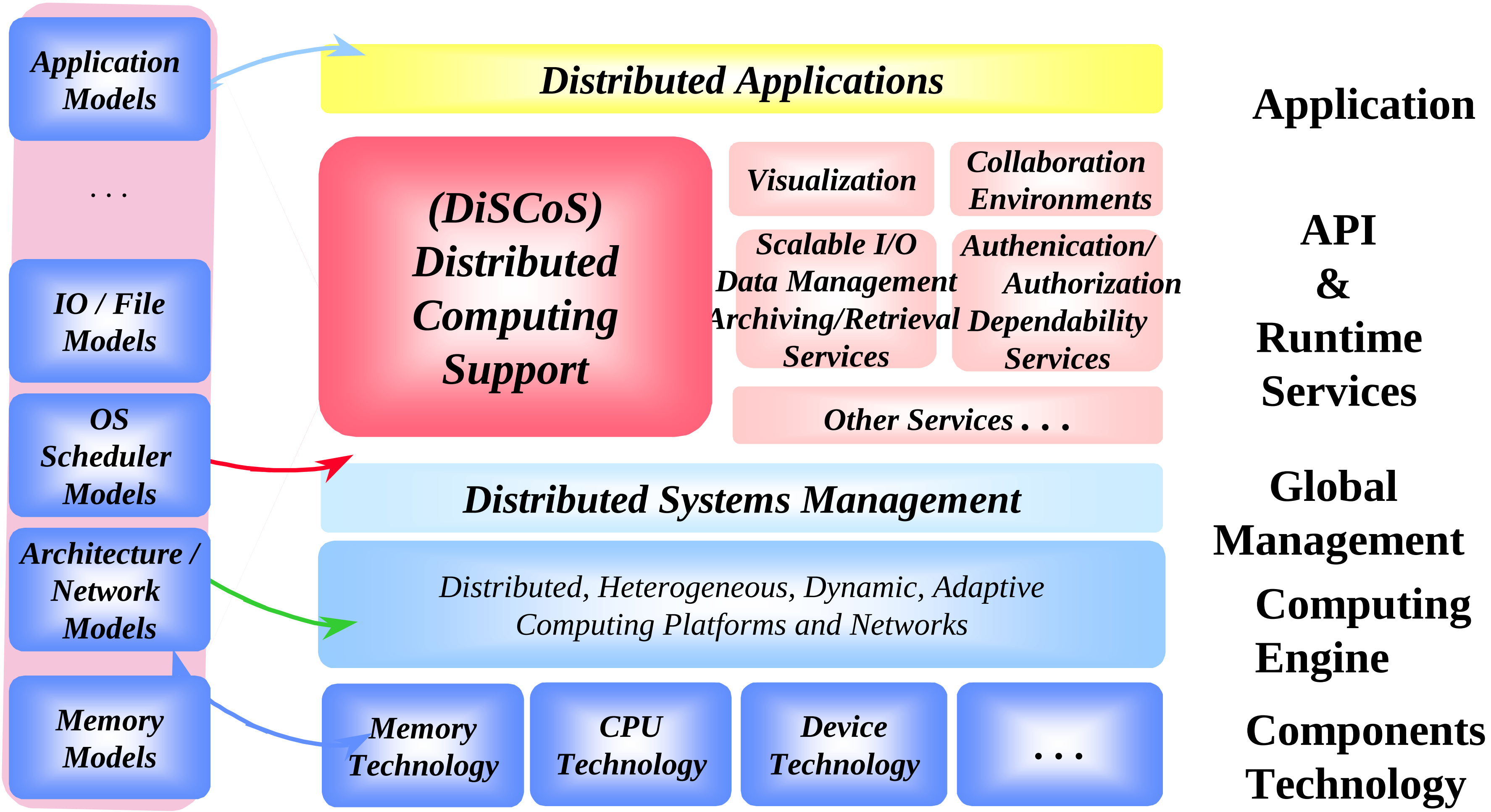
API & Runtime Services

Global Management

Computing Engine

Components Technology

Distributed Systems Software/Hardware Architectural Framework



Application Models

. . .

IO / File Models

OS Scheduler Models

Architecture / Network Models

Memory Models

Distributed Applications

(DiSCoS) Distributed Computing Support

Visualization

Collaboration Environments

Scalable I/O Data Management Archiving/Retrieval Services

Authenication/ Authorization Dependability Services

Other Services . . .

Distributed Systems Management

Distributed, Heterogeneous, Dynamic, Adaptive Computing Platforms and Networks

Memory Technology

CPU Technology

Device Technology

. . .

Application

API & Runtime Services

Global Management

Computing Engine

Components Technology

Application Analysis System Challenges and Approaches

Present methods and tools for performance analysis

- Modeling (queuing and analytical models)
- Simulation tools
  - architecture, network, cache, and I/O simulators
  - trace-driven, execution-driven simulations
- Performance data generation and collection
  - software assists (user directives, libraries)
  - hardware monitors
- On-line analysis and post analysis; Visualization

Problems with present technology

- Existing performance methods and tools study isolated system components
- Interaction of design features across different system layers not well understood
- No means to exploit design information at one level for another level (compiler-architecture for optimization of data mapping, or task scheduling)
- Dynamically-changing heterogeneous systems are even harder to analyse

DisCoS approach for performance analysis:

- Low Risk:
  - **Enable optimizations via application directives to the compiler**
  - **Develop simple parametric models of the application and underlying platform**
- Mid-Risk:
  - **Use parametric application models with system software and hardware models for optimizing task scheduling and partitioning by the compiler**
- High Risk:
  - **Develop performance frameworks with multi-resolution, integrable models across all levels of the system hierarchy, for more accurate compiler optimizations and mapping**