What's Keeping Hard Real-Time Scheduling from Being a Mainstream Technology in the Embedded Multiprocessing Domain Space

Daniel M. Lorts

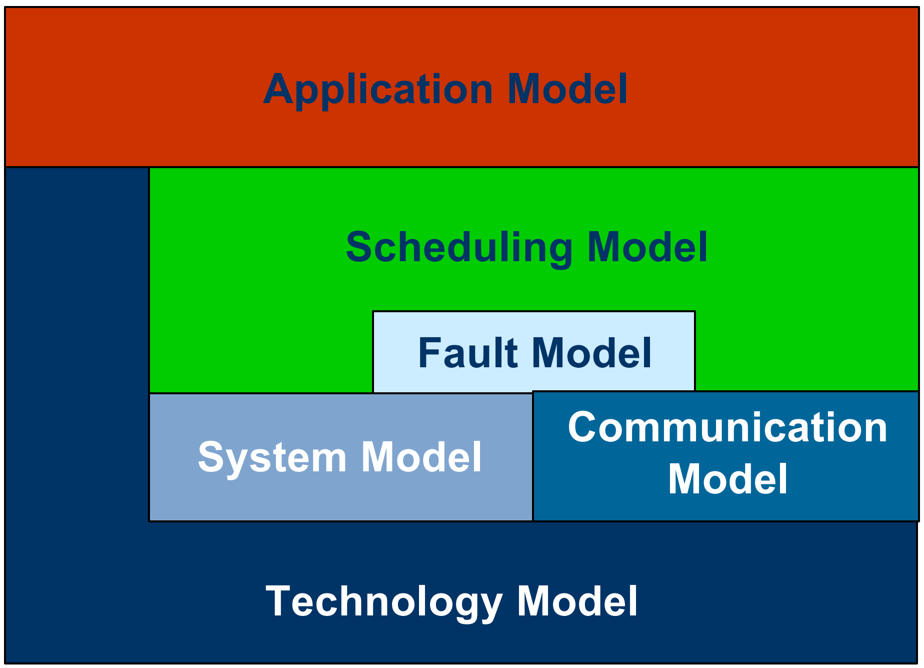
University of Texas – Dallas and Mercury Computer Systems, Inc.

HPEC - September 2002

The Ultimate Performance Machine

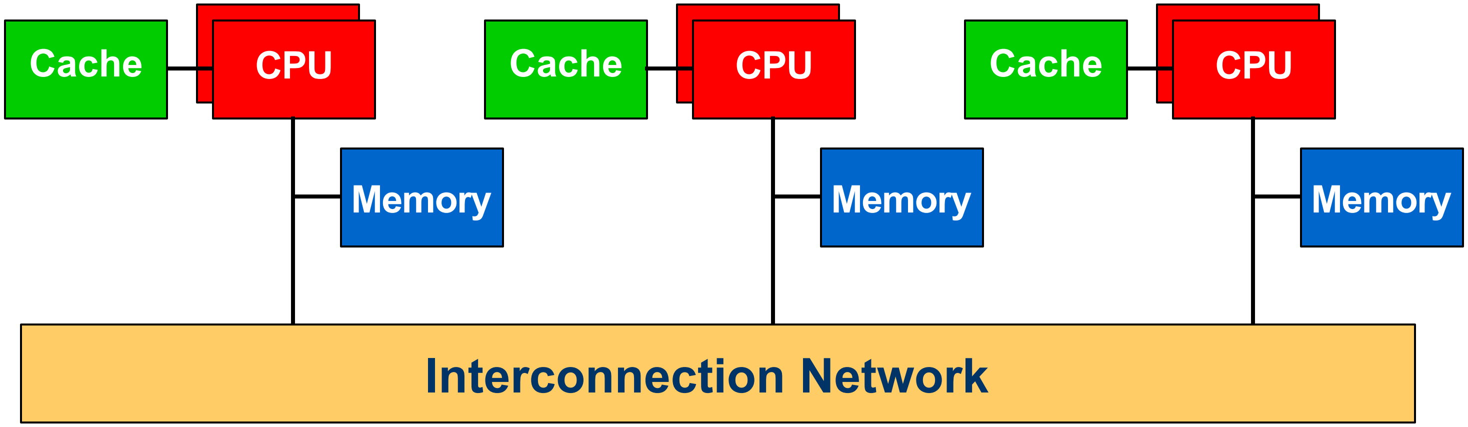
Assumption Model

- The choices and assumptions made in the development of real-time systems affect many areas.
- In this research, we look at six individual, but closely related components of a system architecture.



- To make the scheduling problem simpler, various assumptions, or boundary conditions, in one or more of the models are typically made.
- Why?
  - No-hard/complete problem
  - Single semester projects
  - Limited tenureship of research
  - Focused interest/purpose

System & Communication (current)



- System model assumptions
  - Heterogeneous processing assets
  - High-level processing capabilities
  - Fictitious architectures and topologies
  - Assets fully connected
- Communication model assumptions
  - Negligible communication costs
  - Uniform communication costs
  - Non-contentious communications
  - No priority discernment

Multi-Level System Graph

$S = (R, L)$ top level system definition

$R = (r_1, r_2, \ldots, r_N)$ N resources of system

$\mathbf{L} = \{l_{<à,c>} \mid à \subset R \wedge c > 0 \wedge Adj(à) = 1\}$

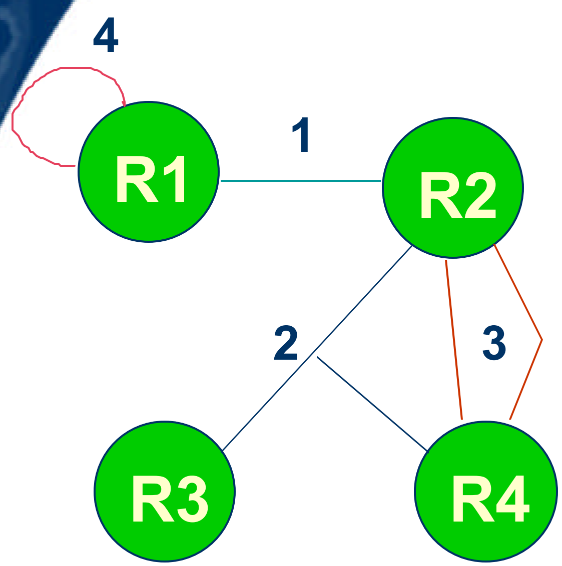
- l represents a link connecting two resources
- à is a subset of the resources of system (R)
- c represents the number of links between the resources
- Adj() is a binary function testing for presence of links

- **Advantages of the MGS:**
  - **Multi-path capability between resources**
  - **Total # of communication links bounded by I/O ports**
  - **Ability to model all multiprocessing topologies**
  - **Scalable to account for resources that have multiple functional units**

    $r_1 = \{c_1, c_2, \ldots, c_M\}$

    where c_j is a unique capability of resource r

An MSG Example



S=(R,L) where

R={R1,R2,R3,R4}

and $L=\{l_{<à1,c1>}, l_{<à2,c2>}, l_{<à3,c3>}, l_{<à4,c4>}\}$

with à1 = {R1} and c1=1

à2 = {R1,R2} and c2=1

à3 = {R2,R4} and c3=2

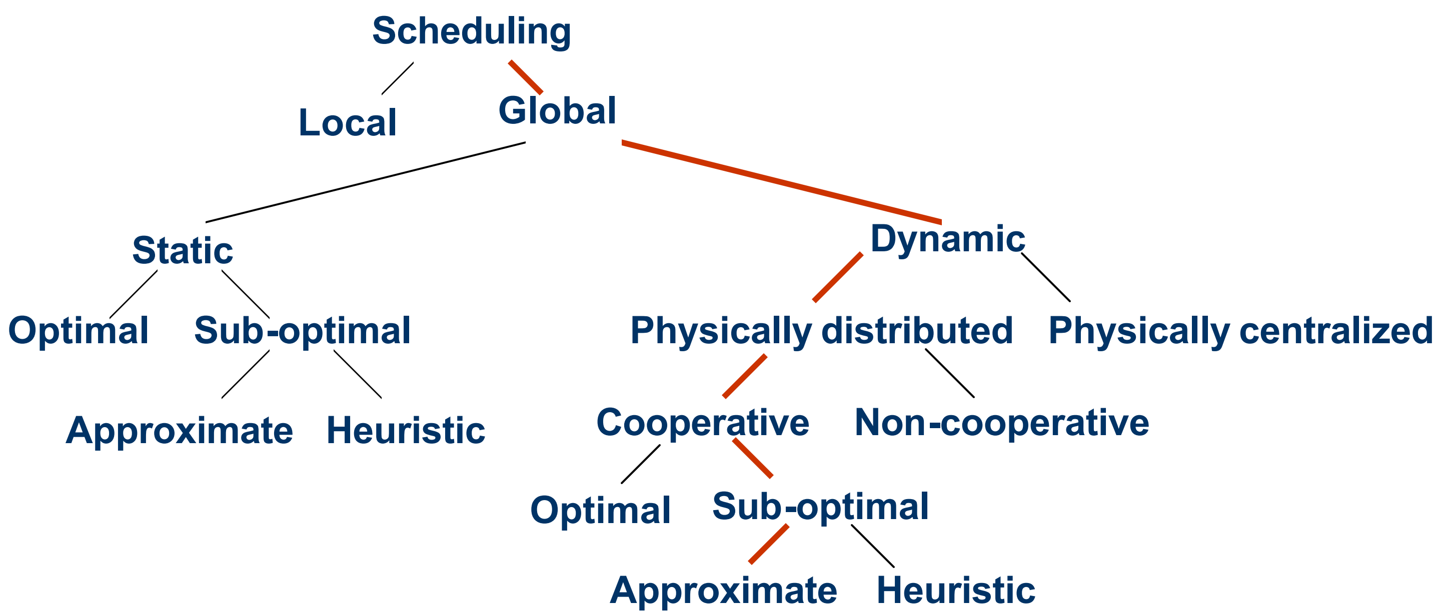
à4 = {R2,R3,R4} and c4=1

	R1	R2	R3	R4
R1	l(4,1)	l(1,1)		
R2	l(1,1)		l(2,1)	l(2,1) l(3,2)
R3		l(2,1)		l(2,1)
R4		l(2,1) l(3,2)	l(2,1)	

Alternate Representation

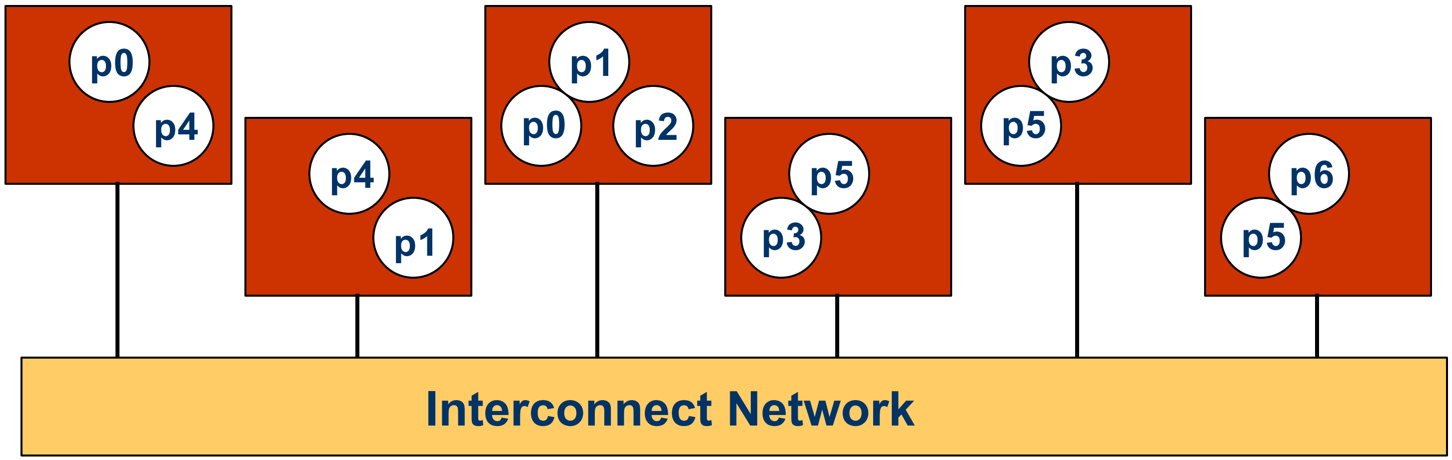
The system model can also be defined mathematically within a table. Each cell represents the paths that exist between each resource : l<id,count>.

Scheduling Model



- **Dynamic scheduling**
  - **Transfer policy**
  - **Selection policy**
  - **Location policy**
  - **Information policy**
- **Heuristics include:**
  - **Heavy Node First (HNF)**
  - **Critical Path Method (CPM)**
  - **Weighted Length (WL)**
  - **Earliest Deadline First (EDF)**
  - **Least Laxity First (LLF)**

Dynamic Framework

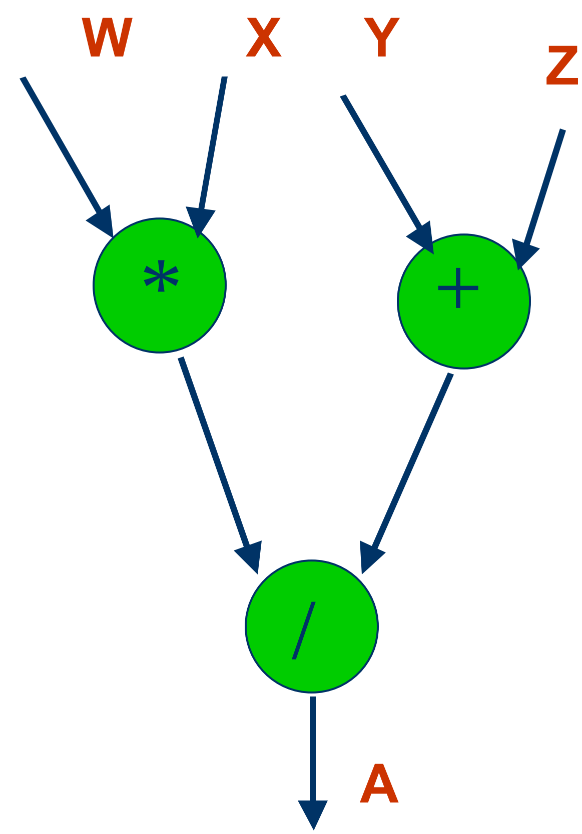


- Allocation stage: statically assigning processes to resources
- Scheduling stage: dynamically based on allocation and application
- Provides run-time analysis of loading and balance
- Scalable solution for all multiprocessing system applications
- Possible multicomputer processor configurations
  - Round-robin/next available
  - Single parallel cluster
  - Pipeline of parallel clusters
  - Hybrid

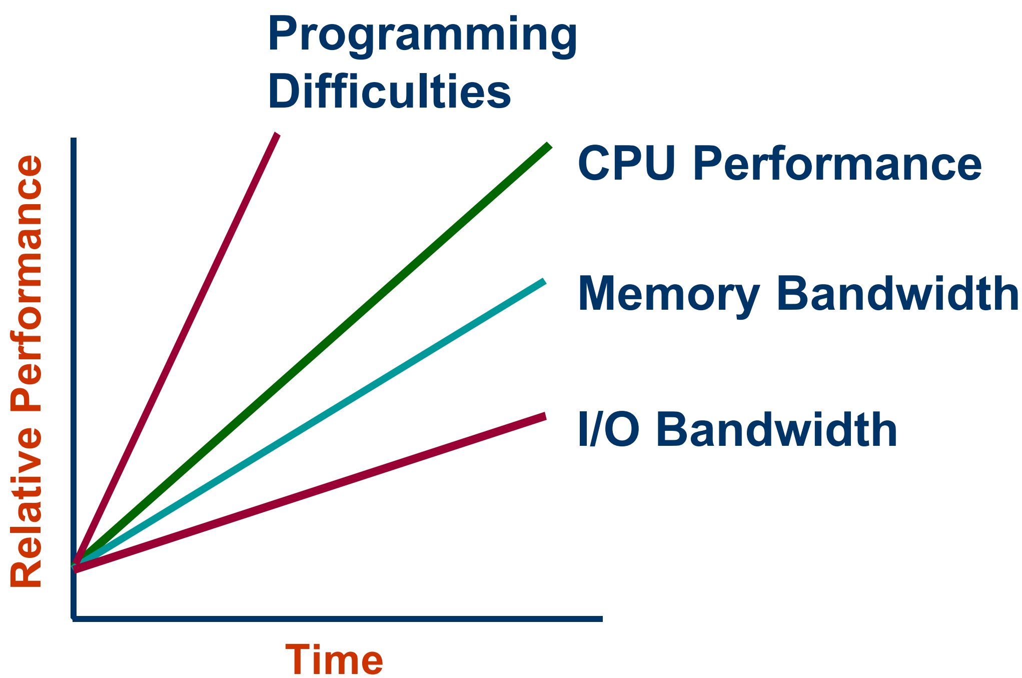
Application Model

- Non-realistic program models (DAGs)
- Task model limited
  - Uniform temporal metrics
  - Preemptability
  - Entry points into nodes
  - Typically unary dependencies
  - Limited methods of prioritization
  - Acyclic models limited
- Task Variables
  - Computational times
  - Communications times
  - Deadlines (laxity)
  - Precedence
    - Number of children
    - Number of parents

<u>Directed Acyclic Graph</u>

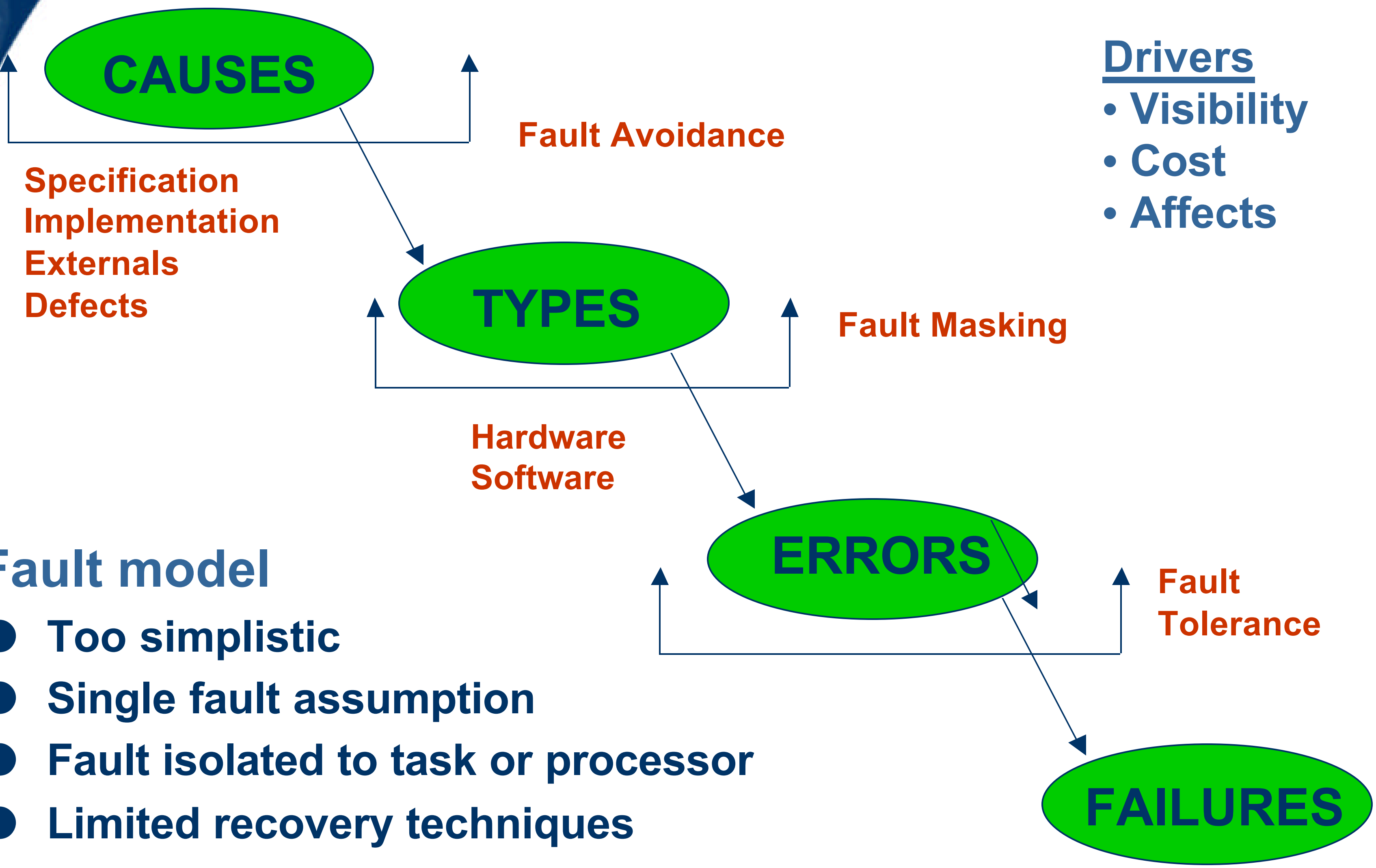


Technology Model



- **Other technological issues**
  - Compiler optimization techniques
  - Multifunctional resources
  - Size, weight, and power considerations

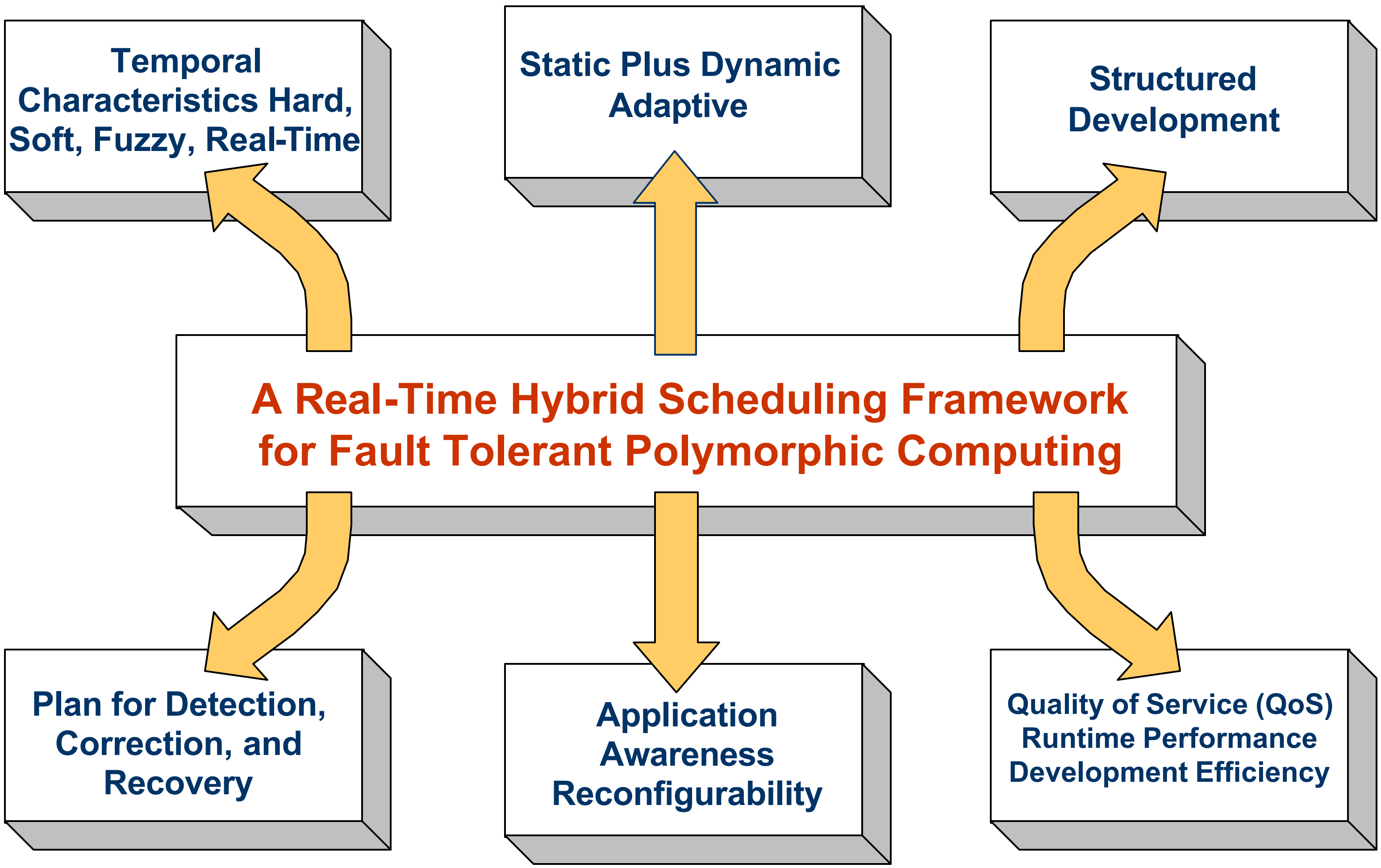
Fault Modeling



Drivers
- Visibility
- Cost
- Affects

- Fault model
  - Too simplistic
  - Single fault assumption
  - Fault isolated to task or processor
  - Limited recovery techniques
  - Inconsistent QoS issues

The Scheduling Framework



Framework Details

- Structured development
  - Provides foundation
  - Validated by mapping know architecture
- Hybrid scheduling
  - Focal point of research
  - Static and dynamic approaches
- Real-Time
  - Formal temporal methods
- Fault tolerance
  - Dynamic detection, correction, and recovery
- Polymorphism
  - Reactive environments
  - Efficient ‘morphing’
- Computing
  - Quality of Service (QoS)
  - Usability and generality