Concurrent Systems Architectures

Kurt Keutzer
EE290A
Wednesdays 2-4PM
Concurrent system architectures

Influencing the concept:
- Software architectures – Garlan and Shaw, Sommerville
- Design patterns – Gamma et al.

Associated issues with description of application/device:
- Granularity of description
- Level of abstraction of description

Related formal issues:
- Formal notions of time and definitions of concurrency
- Development of invariants, preconditions, post-conditions, proofs of correctness

Course goal: Develop a formal notion of concurrent-system architectures as well as a set of CSA’s that help to model the applications and devices that we are looking at.
Applications and Programming Environments

We wish to develop domain specific programming environments.

Domains:

- Network applications
- Wireless applications
- Video applications

Course goal: Evaluate the extent to which CSA’s are useful in the development of a programming environments
Programming Environment: Hypothesis

A successful programming environment is:

• Natural to the application and easy to program in
• Uses the terms/nomenclature (e.g. channel) of the application

Is built from:

• One or more (hierarchical) CSA’s
  • The CSA’s are themselves built on models of computation
• A series of domain specific niceties (e.g. channels, above)
• Extensive application library support – embodying key elements of the applications and the environment it operates in

It offers:

• Opacity: Hiding of implementation detail
• Visibility: Allows for sufficient detail and potential for refinement that any feasible limit of performance can be realized
Applications exploration/presentation format

Application definition/context
Hierarchical block description/concurrent-system architecture/software architecture
  • With a couple/few levels of hierarchy
  • Describe design decisions in refinement
  • Describe key computation components and communication connectors
  • Describe/speculate on models of computation of components and connectors

What libraries would be useful to support this application?
What are the key elements of performance on this application (e.g. maintaining line speed, avoiding congestion etc.)?
Any existing academic/commercial tools and environments that aim to support this class of application – describe them
Applications

- Packet forwarding (Chidamber)
- Voice over IP (Matthias)
- Reed solomon encoding (Matt)
- MPEG-4 encoding & decoding (Kaushik)
We wish to develop programmable platforms-specific programming models.

Domains:

- Network processors
- Communication processors
- Video processors

Course goal: Evaluate the extent to which CSA’s are useful in the development of a programming models.
Programming Model: Hypothesis

A successful programming model:
• Makes the platform easy to program
• Exposes the relevant elements of the underlying device/platform in a natural way

Is built from:
• One or more (hierarchical) CSA’s
  • The CSA’s are themselves built on models of computation
• Naturally abstracts up the hardware support in a application domain specific way
• Offers extensive application hardware support for the key computational kernels of the target applications

It offers:
• Opacity: Hiding of device details
• Visibility: Allows for sufficient visibility and path for refinement that the full performance of the platform can be realized
Approaches to Programming Models

Library component assembly – e.g. COSSAP/SPW DSP

Programming language – e.g. ASM/DSP, MATLAB/DSP
Platform prog model presentation format

Platform definition/context
Hierarchical block description/concurrent-system architecture/software architecture
- With a couple/few levels of hierarchy
- Describe design decisions in refinement
- Describe key computation components and communication connectors
- Describe/speculate on models of computation of connectors
Any existing academic/commercial programming tools, models, environments that aim to support this platform?
Platforms

Nexperia, 2Wire – Kees
Calisto – Gary Banta
Cport - Chidamber
IXP1200 – Niraj
Scott – RAW
Mihal – Flash
iWarp – Yujia
VIP – Christian
Score – Eylon Caspi
Mapping

Mapping is the translation of an application, written in a programming environment, onto the target platform, described by its programming model.

Course goal: Evaluate the extent to which CSA’s are useful in the mapping from programming environments of our target applications onto programming models of our target platforms.
Mapping: Hypotheses

H1: The only successful mapping will be an identity mapping
• Either the programming environment must in fact offer a programming model for the platform OR the programming model for the platform is an acceptable programming environment for the application domain
• If not, the real mapping problem across distinct CSA’s and MOC’s is intractable and even if it is created will never offer the correct level of performance

H2: A theory of mapping from one CSA to another CSA must be developed
  • Mapping from distinct programming environments onto distinct platforms IS possible, but additional theory is required
  • Key element is to map from one CSA to another and one MOC to another

Course goal: To come to conclusions about these hypotheses
Related Mapping questions:

Describe (high-level) mapping of one application onto one architecture
  - How flexible are the CSA’s of the chosen architecture?
  - How do you map from one CSA to another?
    - What’s the theory?
    - What’s the practice?
  - What do you do about concurrency? How do you manage translation from one MOC to another?
  - How do you estimate/observe/improve performance within a CSA?
Mapping projects

Applications
• Packet forwarding (Chidamber)
• Voice over IP (Matthias)
• Reed solomon encoding (Matt)
• MPEG-4 encoding & decoding (Kaushik)
• MPEG2 decode + graphics (Kees)

Architecture:
• Nexperia, 2Wire – Kees
• Calisto – Gary Banta
• Cport - Chidamber
• IXP1200 – Niraj
• Scott – RAW
• Mihal – Flash
• iWarp – Yujia
• VIP – Christian
• Score – Eylon Caspi

Teams of pairs experiment with mapping one application onto one platform