CnC:
A Dependence Programming Model

Kath Knobe kath.knobe@rice.edu

Zoran Budimlić zoran@rice.edu

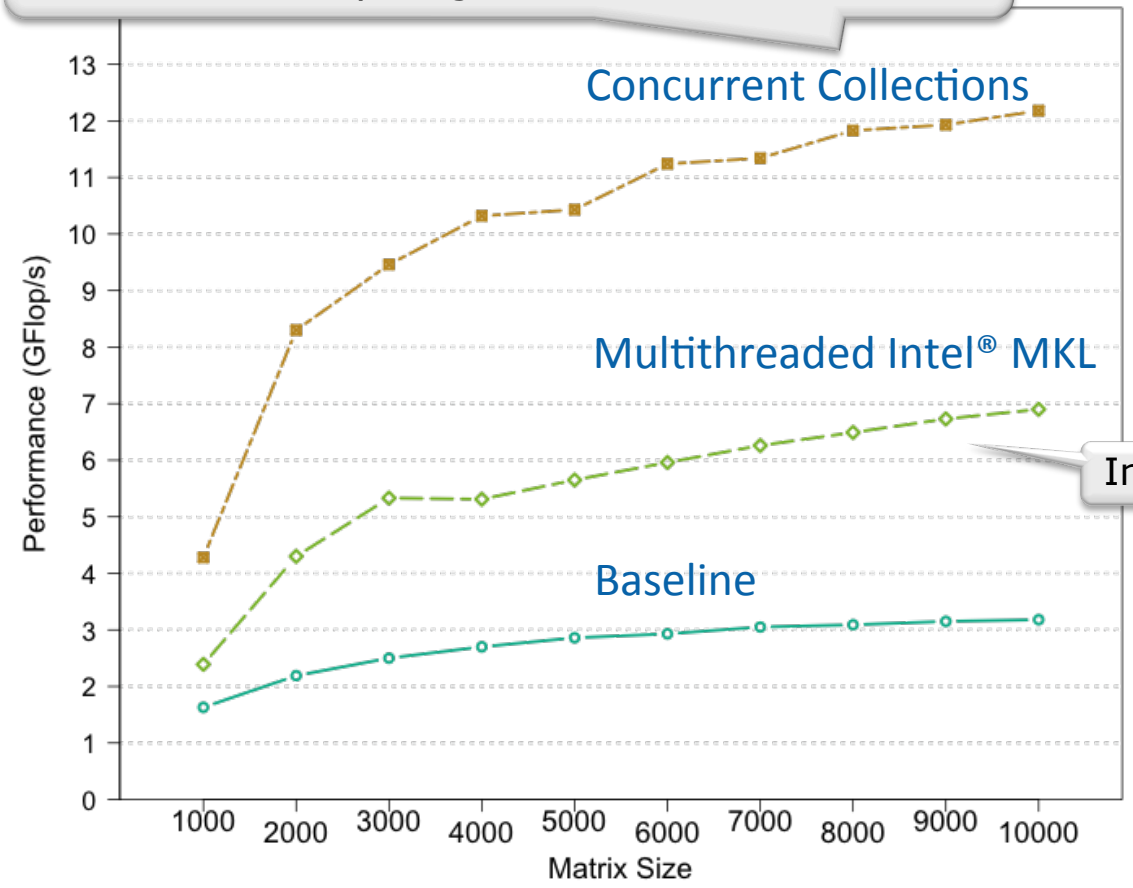
Motivation for the talk (not motivation for CnC)

- Someone hears about CnC
  - To learn more they check into an implementation
  - It wasn't designed for what they want
  - They walk away
- Our marketing department needs to make some changes to how we present/write/... about CnC
- Here's a possible different slant
- You know most of this but I'm looking for
  - Feedback/discussion on the problem
  - Feedback/discussion on the story here
  - Recommendations for changing the presentation
- This seems like the right group

Performance

Eigensolver

Aparna Chandramolishwaren
One GaTech first year grad student intern at Intel



Intel 2-socket x 4-core Nehalem @ 2.8 GHz + Intel® Math Kernel Libraries 10.2

Used on

DARPA:

UHPC project at Intel

DOE:

X-Stack project at Intel

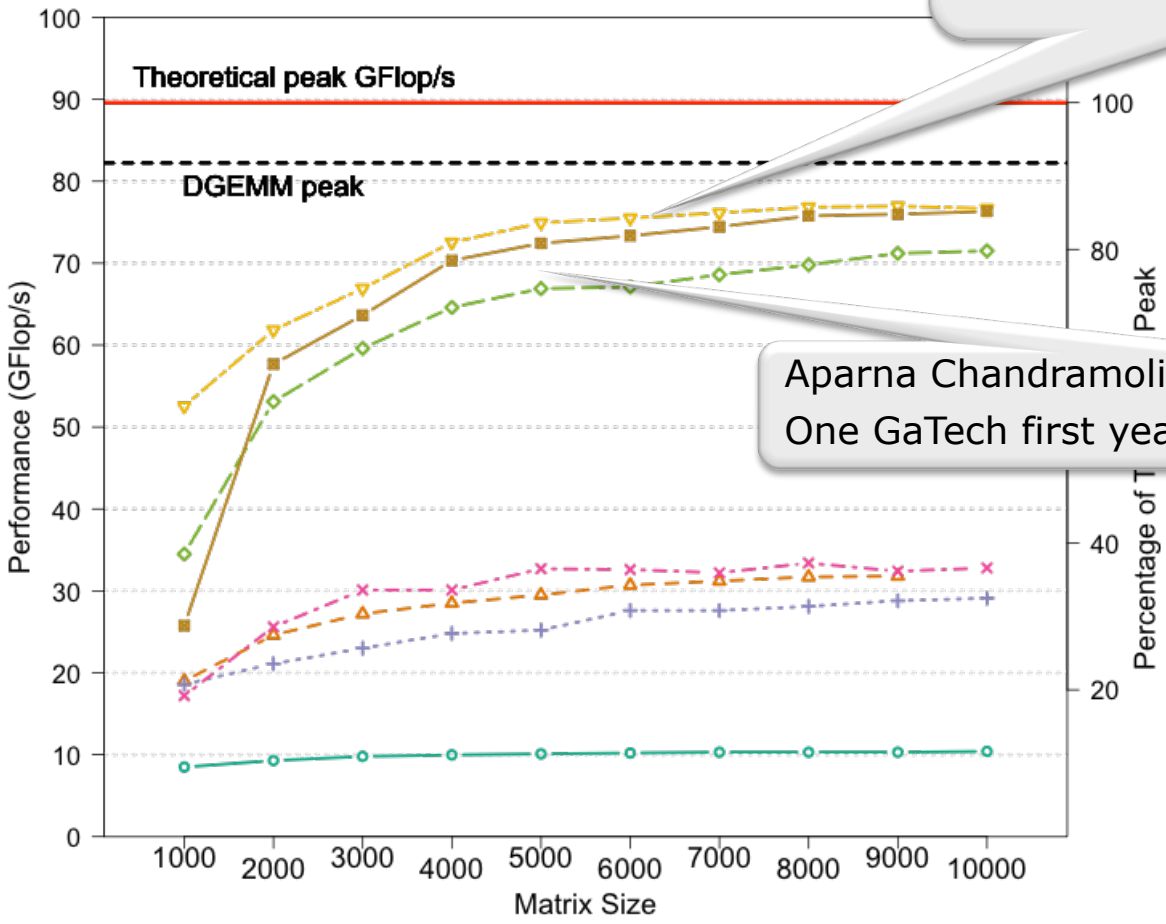
Dreamworks:

"How to train your dragon II"

CnC

Choles

Plasma: Jack Dongarra's group at Oak Ridge NL



Aparna Chandramolishwaren

One GaTech first year grad student intern at Intel

Intel 2-socket x 4-core Nehalem @ 2.8 GHz + Intel MKL 10.2

Outline

1. Motivation
2. CnC
3. Wide array of existing CnC tuning approaches

Outline

1. Motivation
2. CnC
3. Wide array of existing CnC tuning approaches

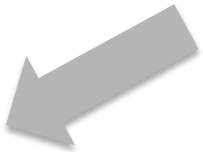
Styles of languages/models

- Serial languages and Par
  - Orderings:
    - Required / tuning / a
    - Make a modification
  - Takes time
  - Leads to errors
    - Too few dependence
    - Too many dependenc
- Dependence languages
  - Orderings:
    - Required only
    - No tuning orderings
    - No arbitrary orderings
  - Make a modification => the ordering requirements are clear
    - No time
    - No errors

- CnC is a dependence programming model
- The dependences include data and control dependences
  - Both are explicit
  - They are at the same level
  - They are distinct

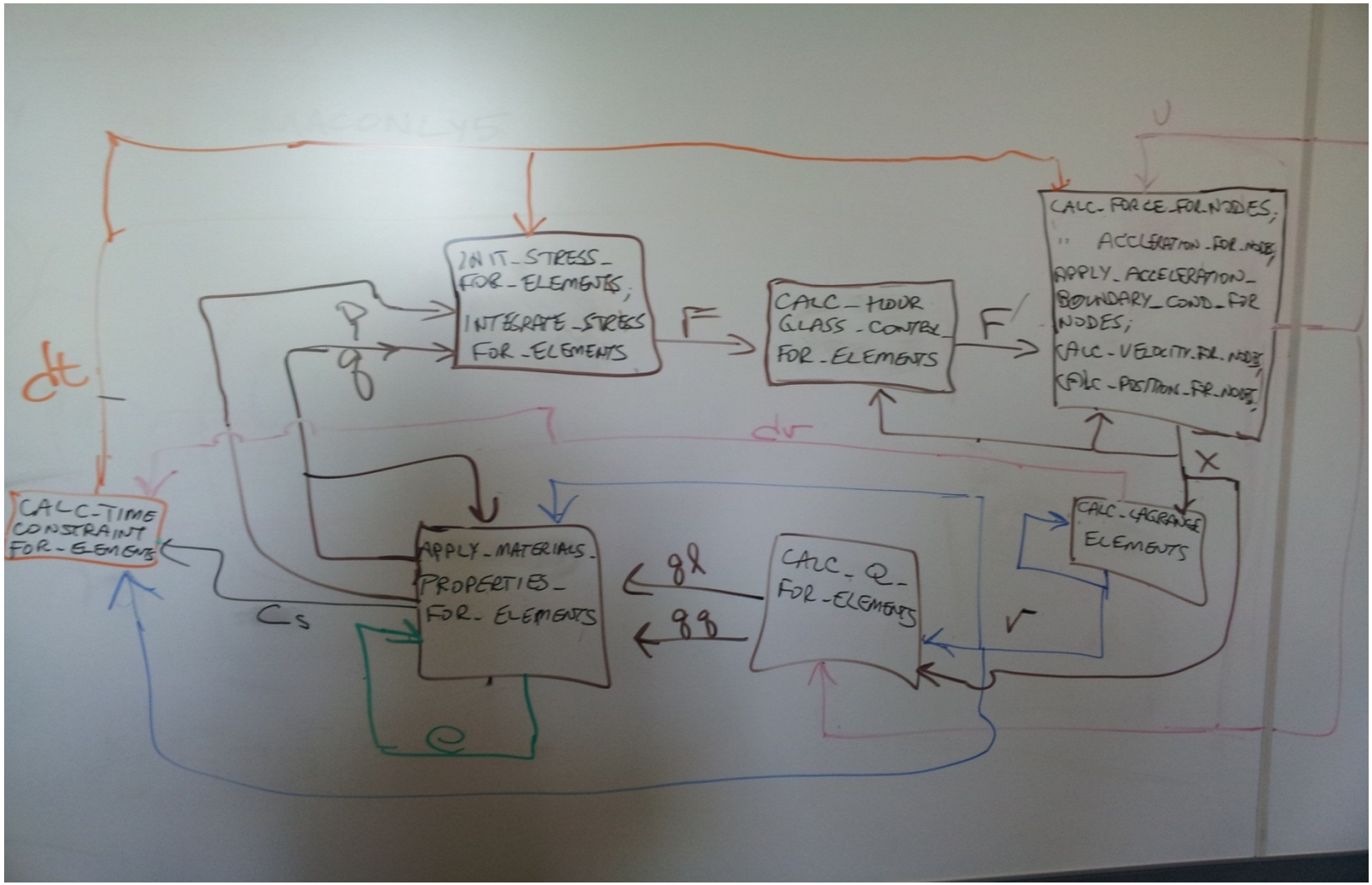
Some relevant current approaches

- Start with an explicitly serial or an explicitly parallel program. Modify it for some specific target or specific optimization goal
  - Have to undo or trip over earlier tunings
- Start with source code and automatically uncover the parallelism
  - We've been there
- Start with the white board drawing
  - It's how we think about our apps
  - Dependences are explicit but: not executable

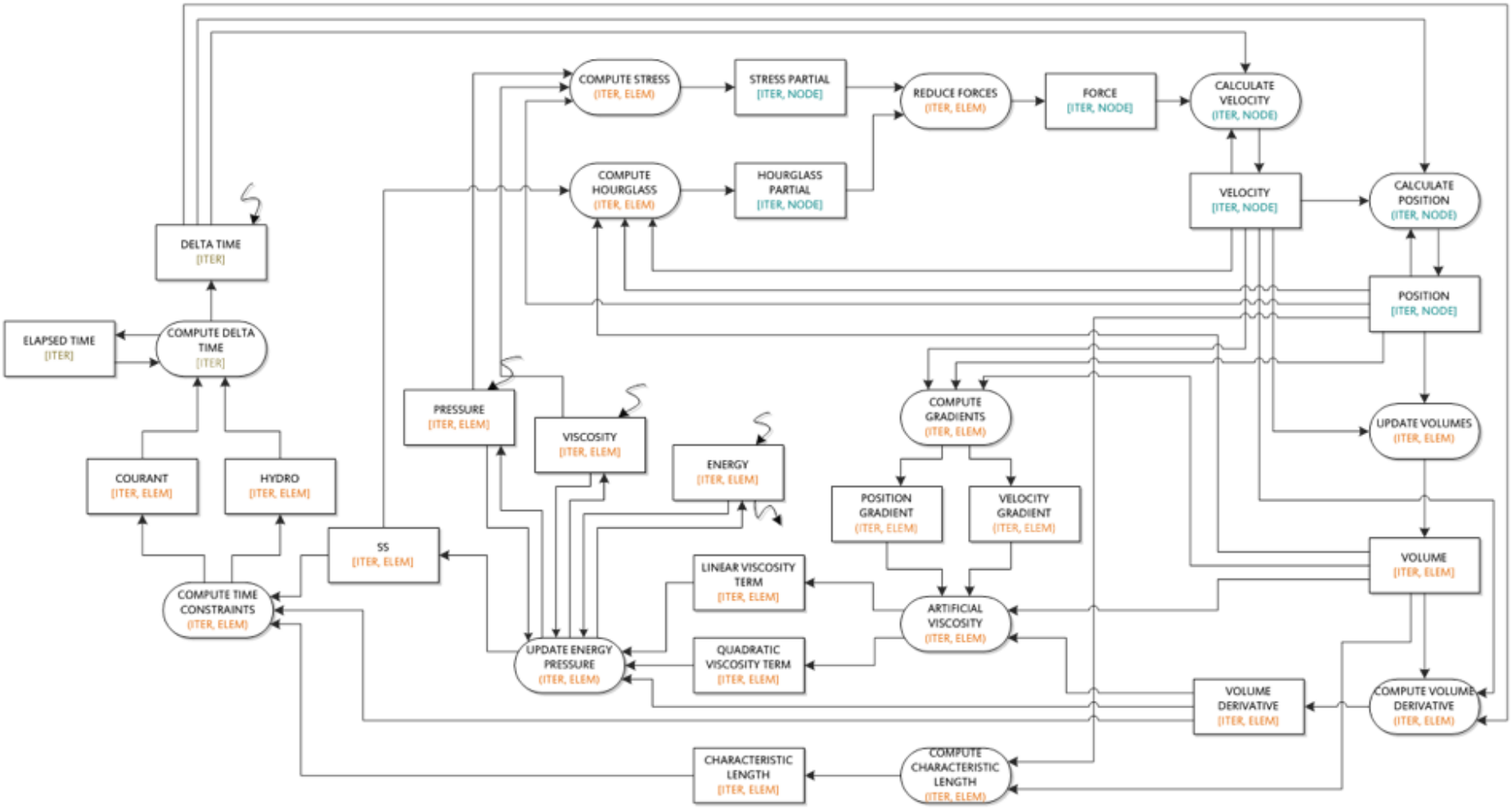


An actual white board sketch: LULESH

a shock hydro-dynamics app



A CnC graph of flow of data for LULESH



Unlike an actual white board sketch CnC is a formal model

- It is precise
  - It is **executable**
- Has enough information
  - To **prove** properties about the program
  - To **analyze** the program
  - To **optimize** the program
- At the graph level
  - Without access to computation code

Outline

1. Motivation
2. CnC
   - Goals that support exascale needs
   - CnC details via an example app
3. Wide array of existing CnC tuning approaches

Needs for exascale

Domain expert
(physicist, economist, biochemist, …)

Tuning expert
(computer scientist).

Software engineering

- Separation of various activities (ways of thinking)
  - The details of the computation within the computation steps
  - The dependences among steps
  - The runtime
  - The tuning
- Leads to im
- Domain exp
  to know abo
  - Architectu
  - Tuning goals

Tuning

- Given only dependences
- Hides irrelevant low level computation details
- Each new tuning starts from just dependences
- Use any style of tuning

  Just obey the dependences

  doesn't

  ut

  cs,

Same or different people
Different activity
Communicate at the level of the graph
Not about physics or about parallel performance

Tuning

- Tuning consumes bulk of the time and energy
- There is no CnC-specific approach to tuning
  - CnC is not a parallel programming model
- Only one requirement:
  - the tuned app must obey domain spec semantics
- Single domain spec / A wide range of tuning goals & styles
  - Faulty components
  - Distinct tuning goals (time, memory, energy, …)
  - Many different styles of runtimes (static / dynamic)
  - Many different styles of tuning (static / dynamic)
- Domain spec isn't modified by tunings
  - Tunings may refer to domain spec but are isolated from it

CnC simplifies tuning
by separation of concerns

- Separates the details of the computations and data from the dependences among them

- Domain spec is isolated from tuning
  - It explicitly represents the ordering requirements
  - No orderings for tuning & no arbitrary orderings

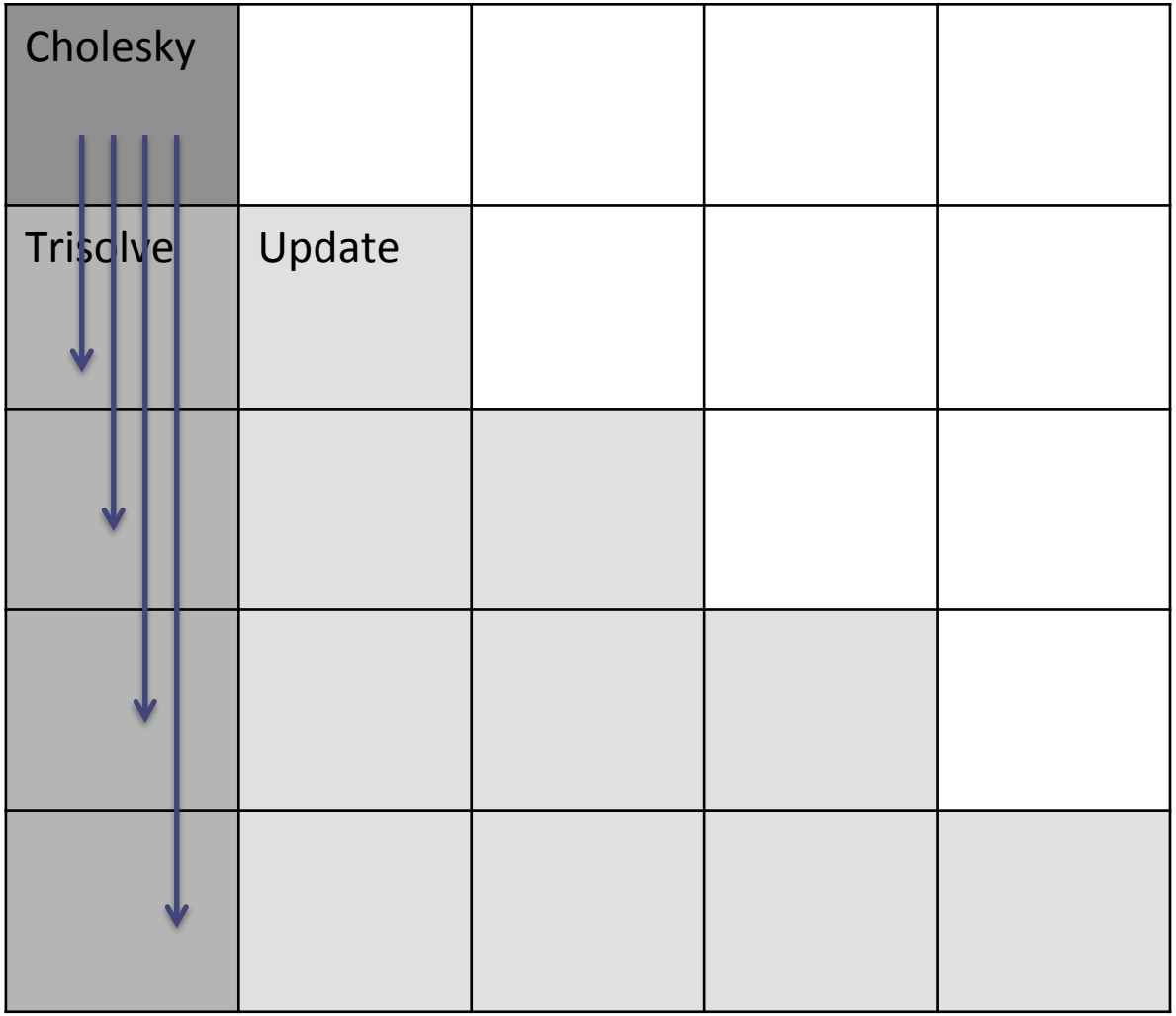
Outline

1. Motivation
2. CnC
   - Goals that support exascale needs
   - CnC details via an example app
3. Wide array of existing CnC tuning approaches

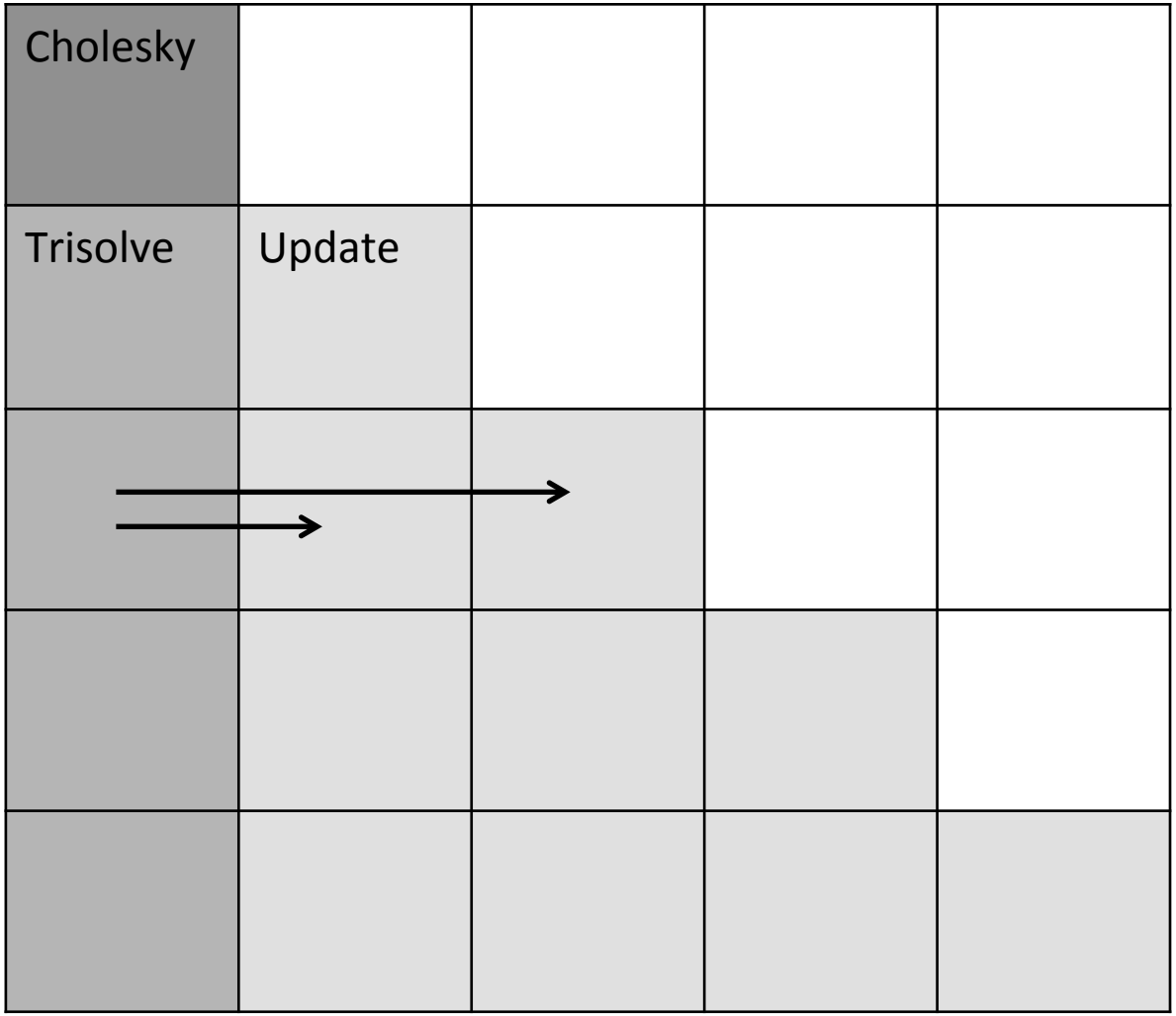
Cholesky factorization

Cholesky				
Trisolve	Update			

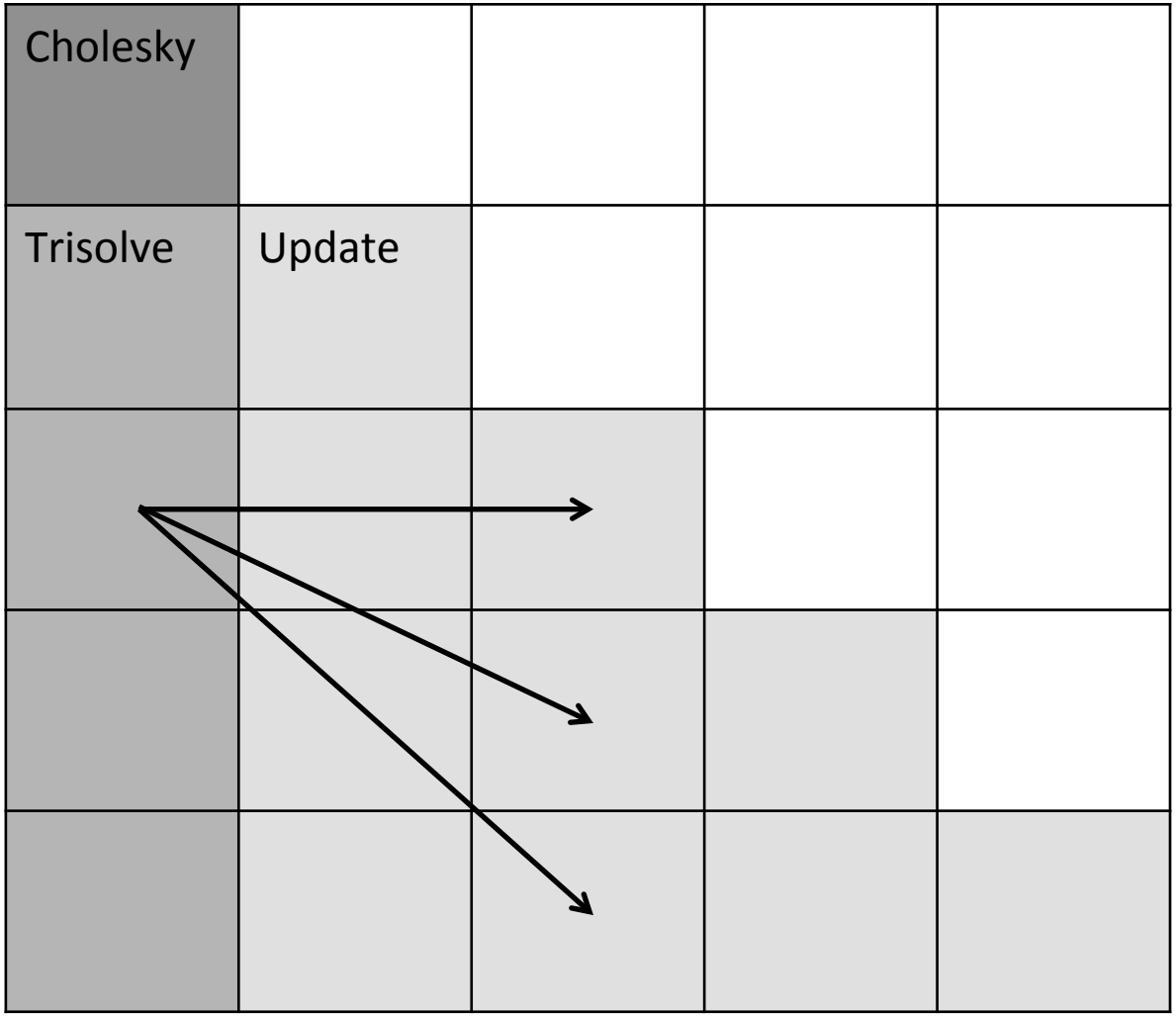
Cholesky factorization



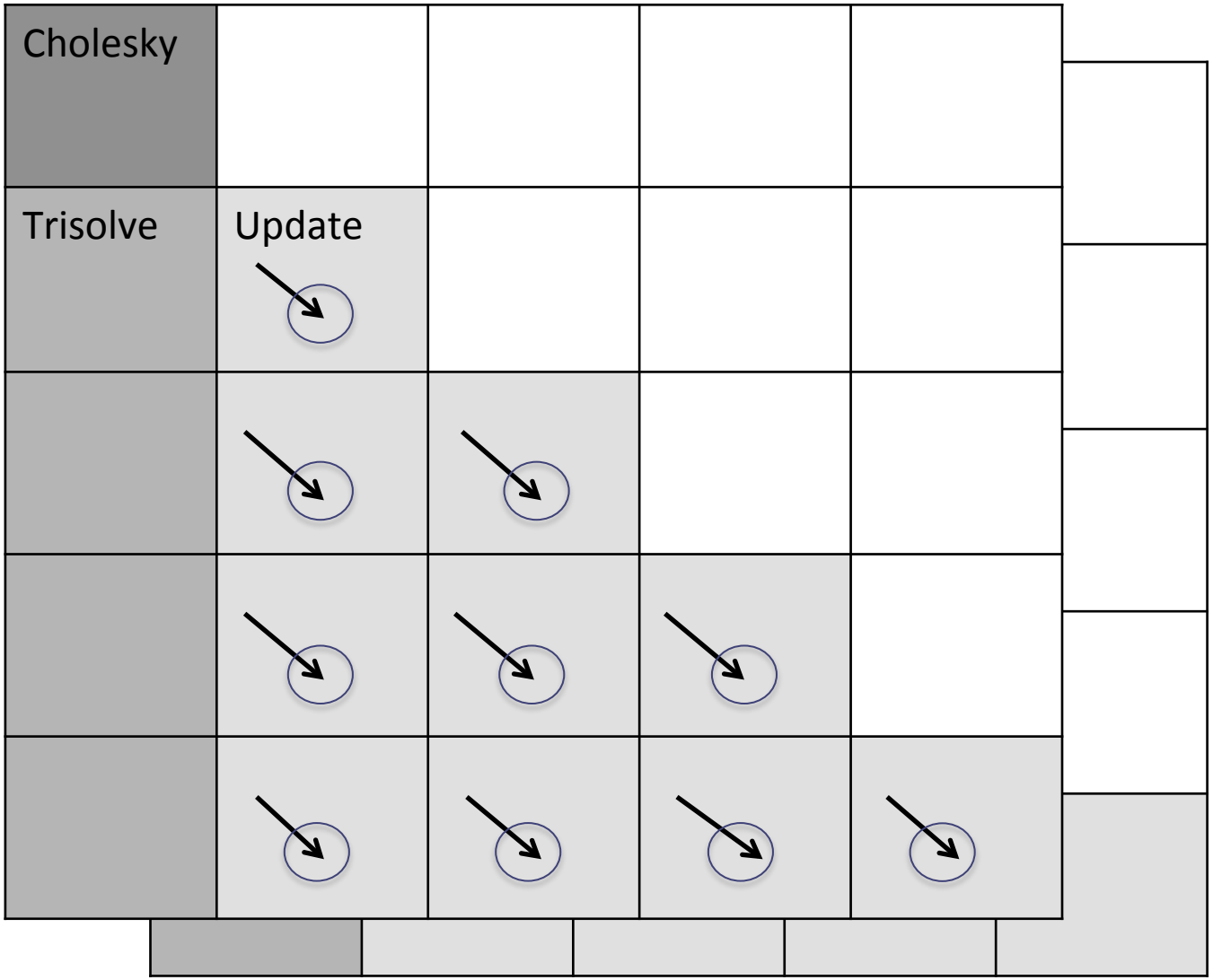
Cholesky factorization



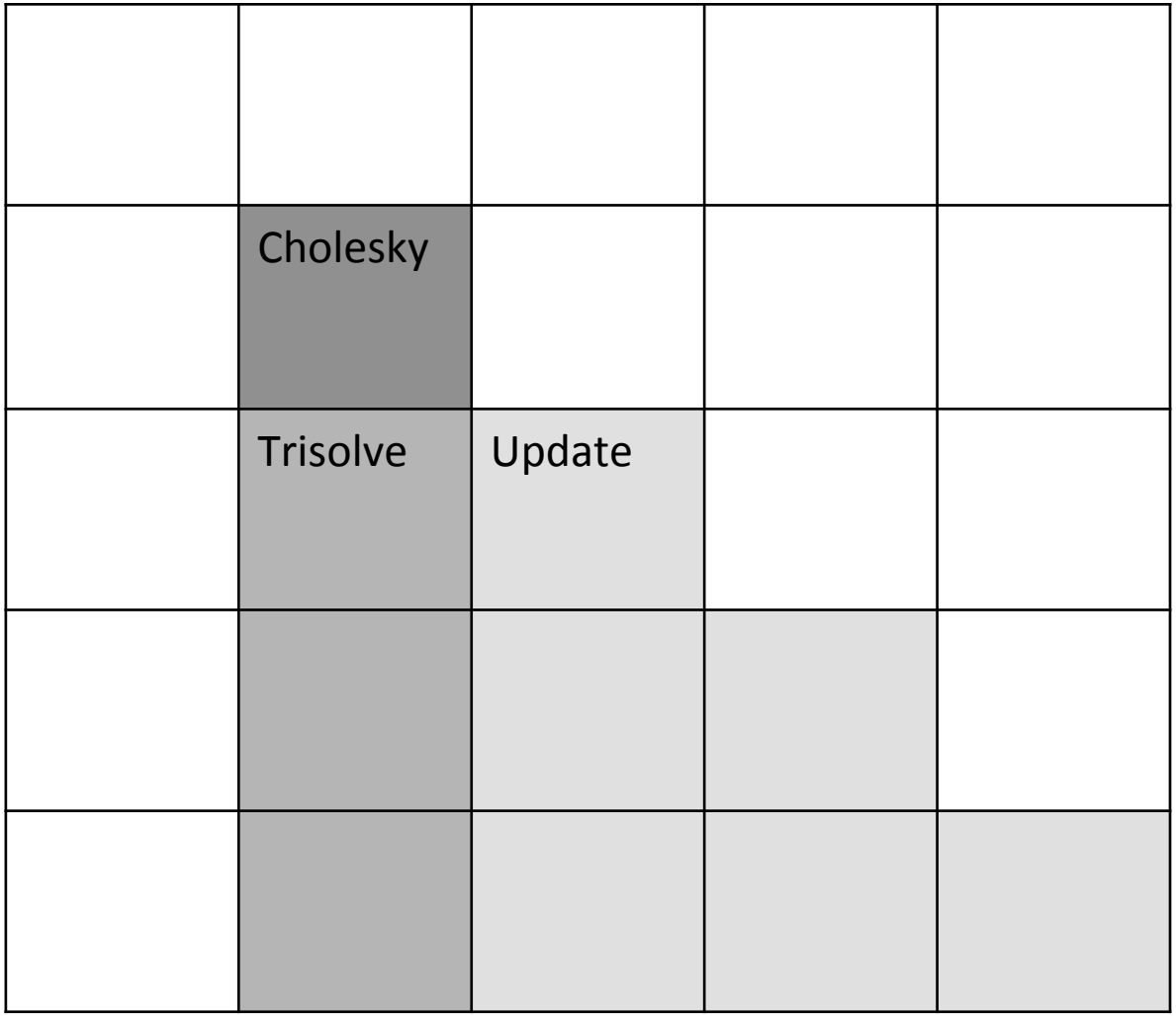
Cholesky factorization



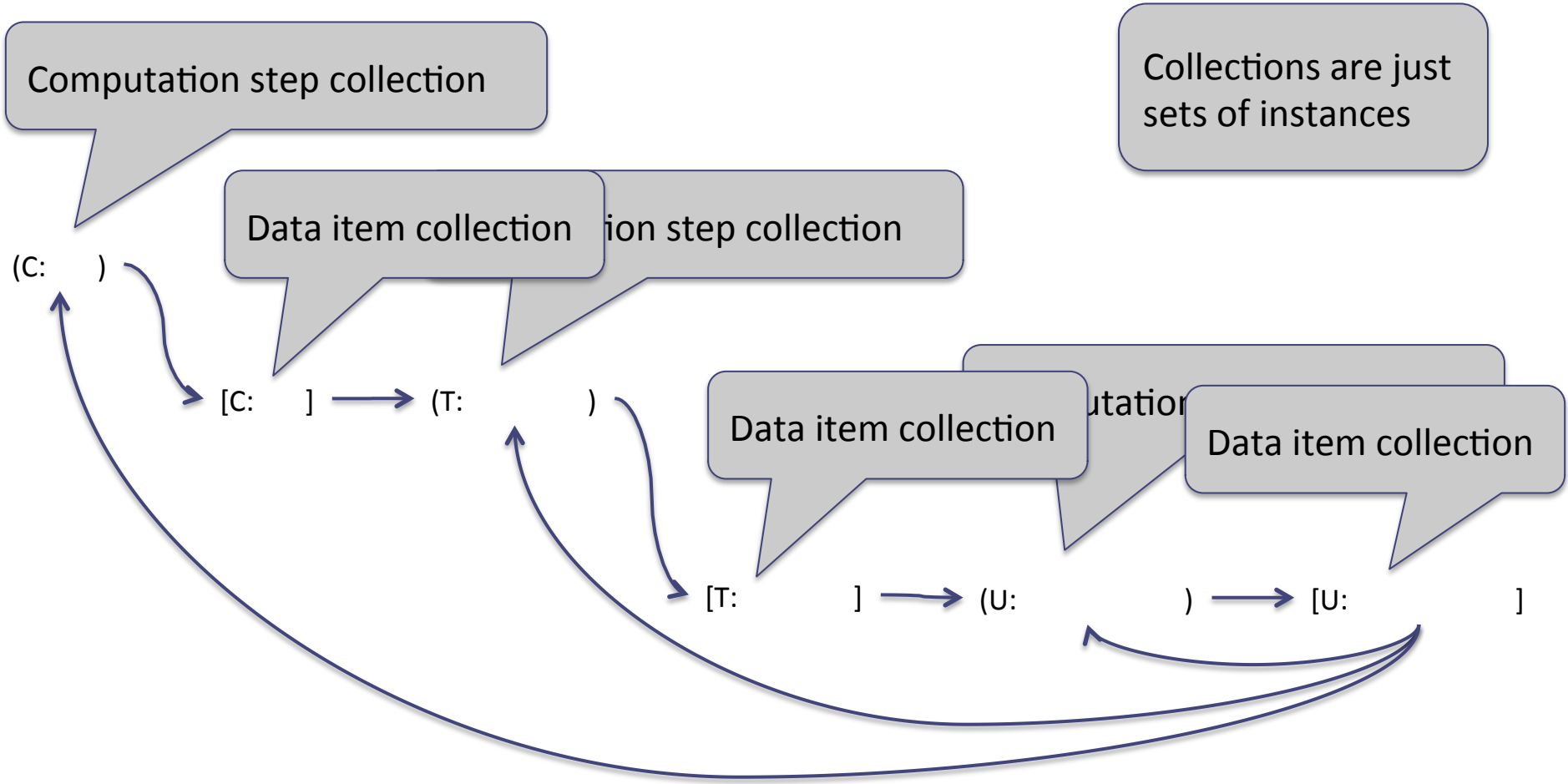
Cholesky factorization



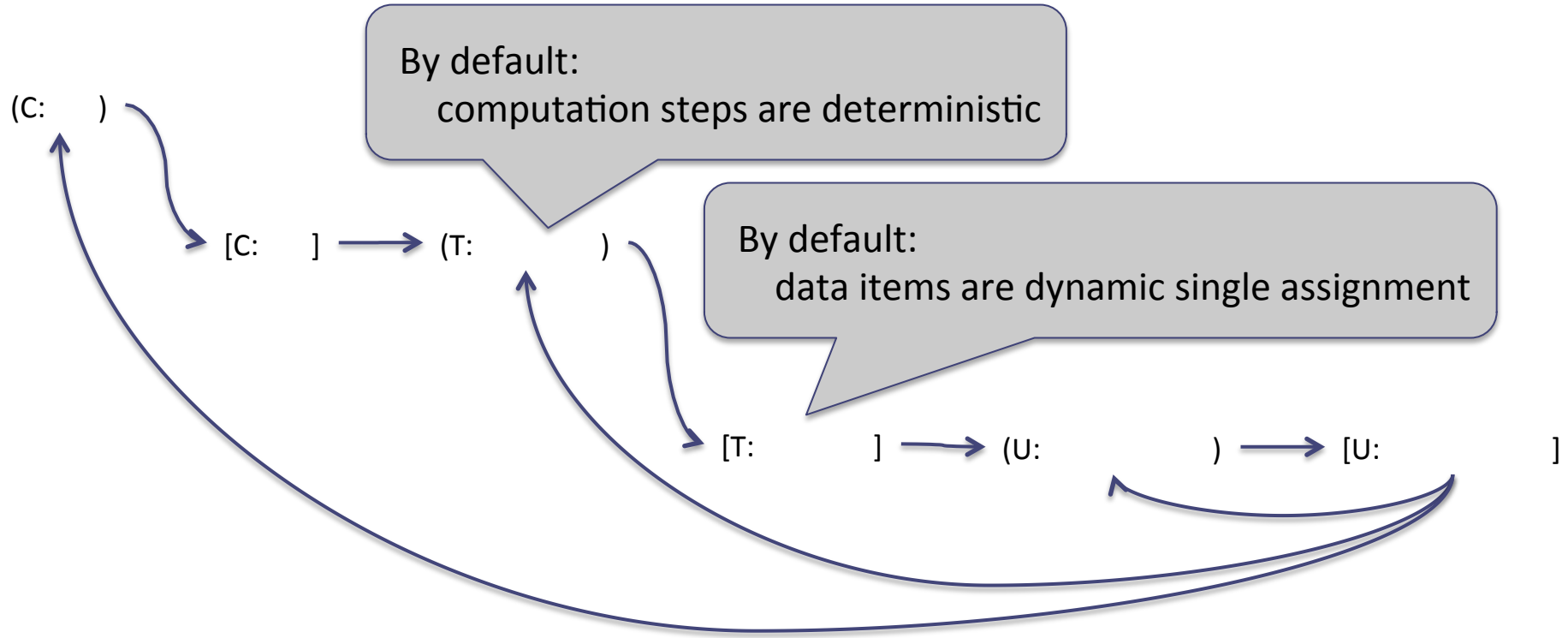
Cholesky factorization



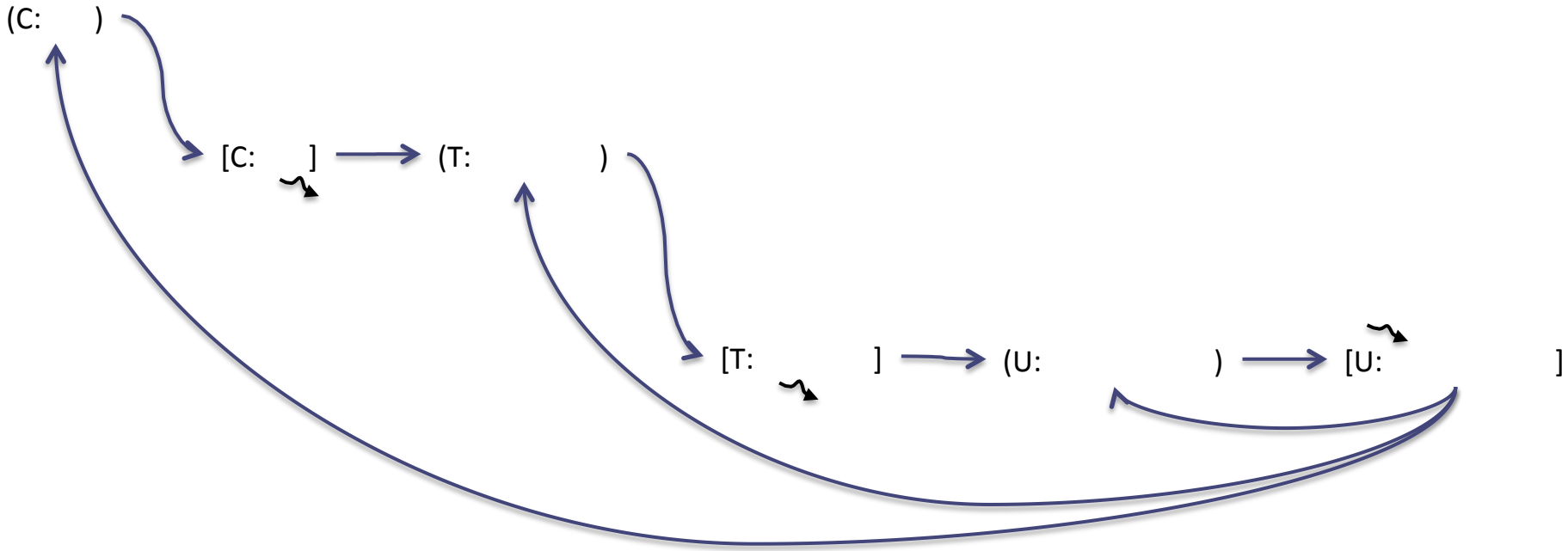
Cholesky CnC domain spec
1: white board drawing



Cholesky CnC domain spec 1: white board drawing

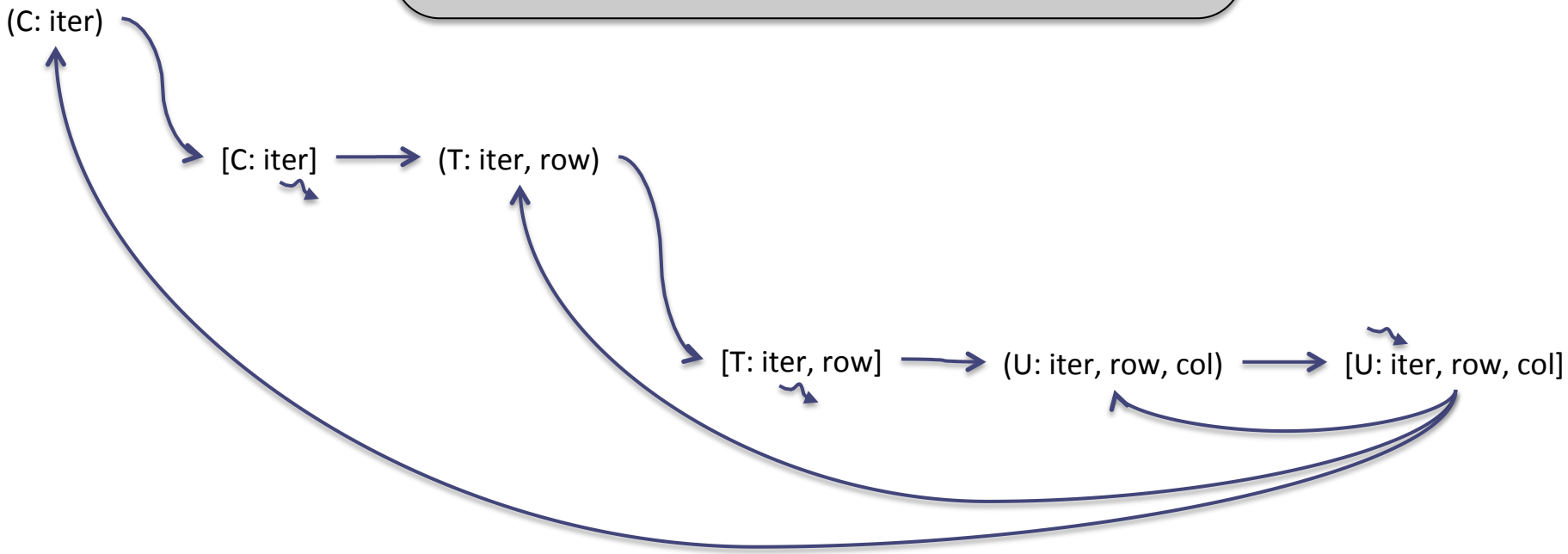


Cholesky CnC domain spec
2: I/O



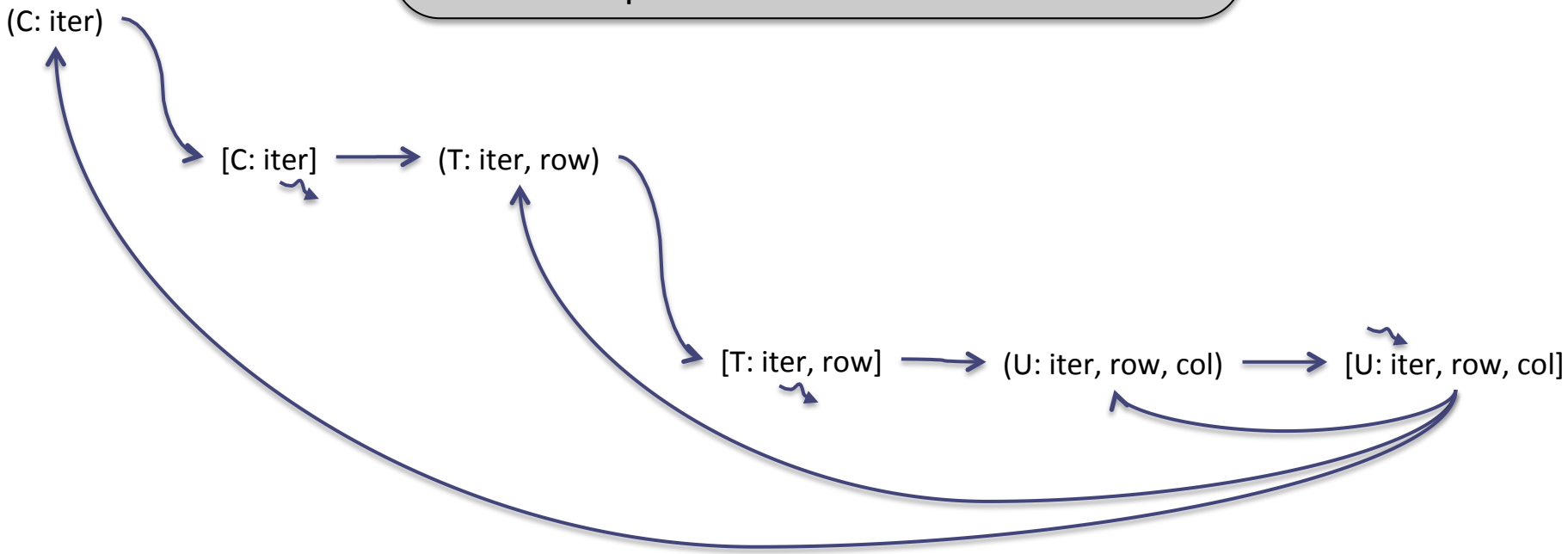
Cholesky CnC domain spec 3: Distinguish instances

Tags are just identifiers
- Require an equality operator
- Often integers: iteration, row, col

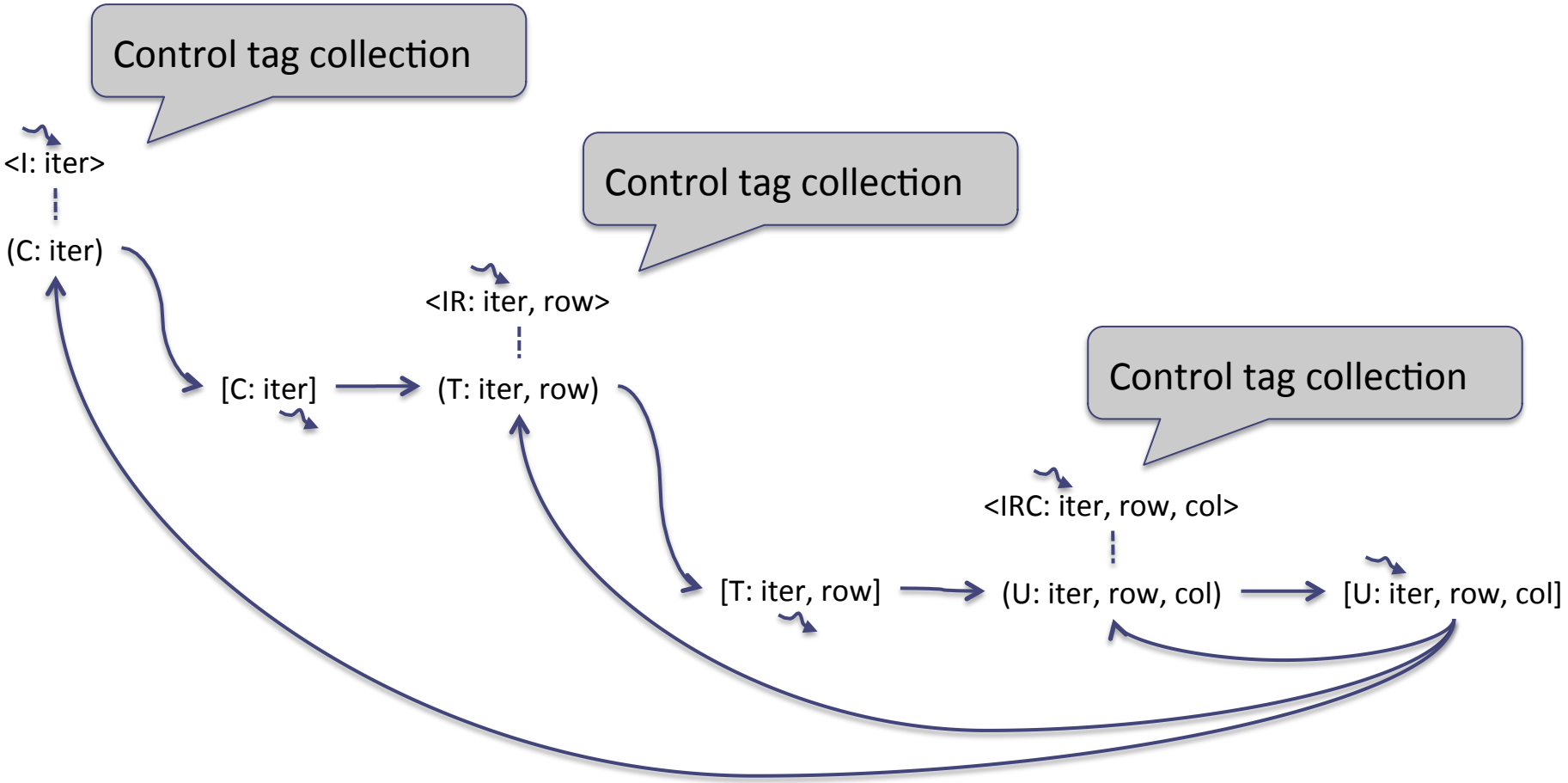


Cholesky CnC domain spec 3: Distinguish instances

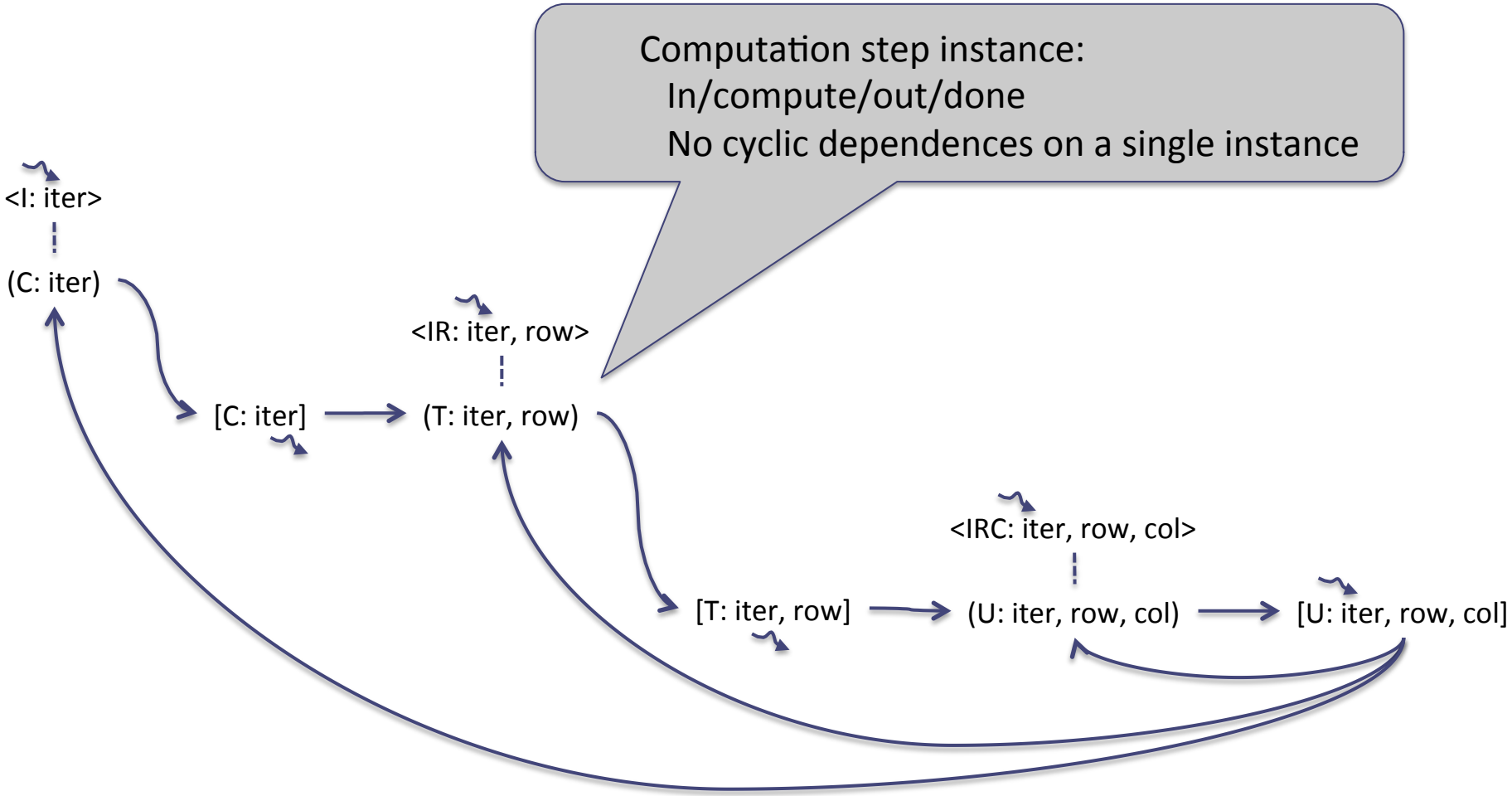
Tags are just identifiers
- Require an equality operator
- Often integers: iteration, row, col
- But not necessarily:
  - representation of a Sudoku board



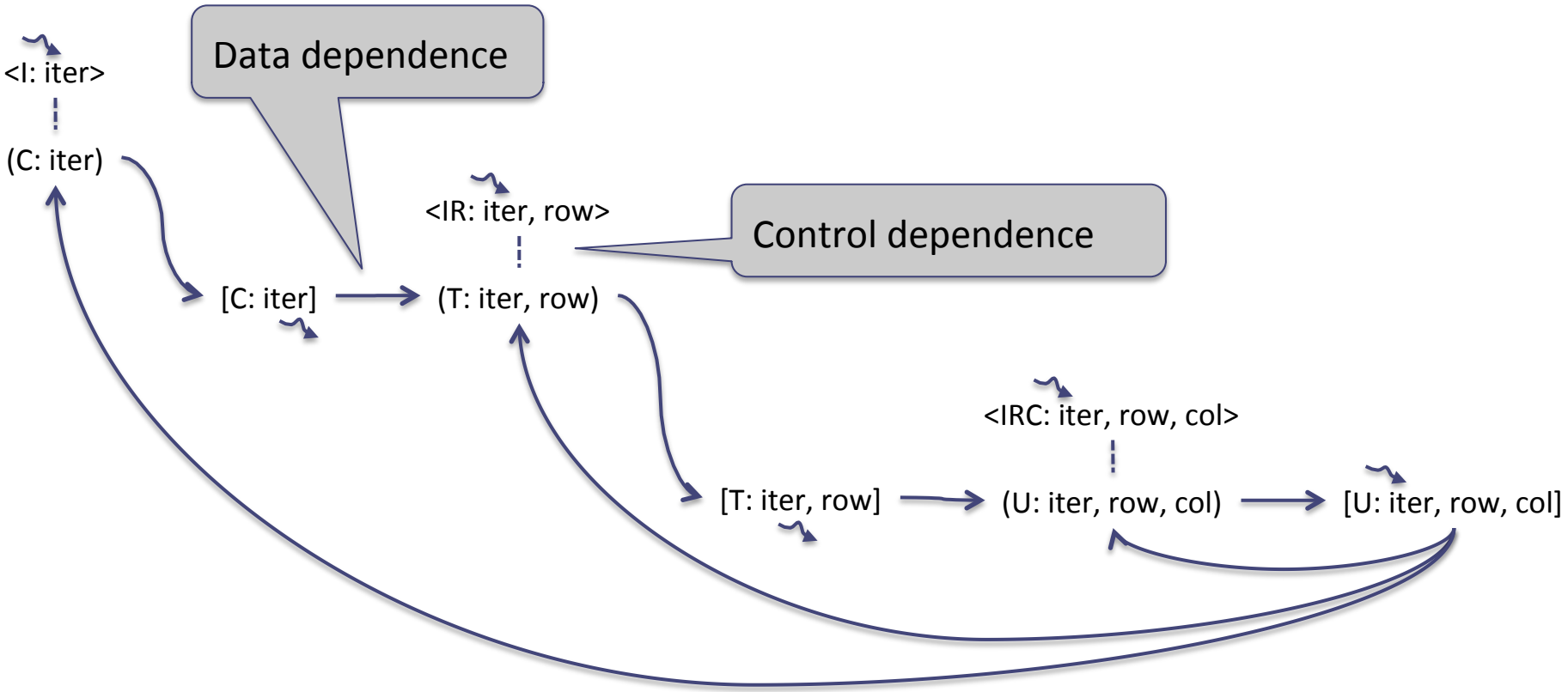
Cholesky CnC domain spec
4: exact set of instances (control)



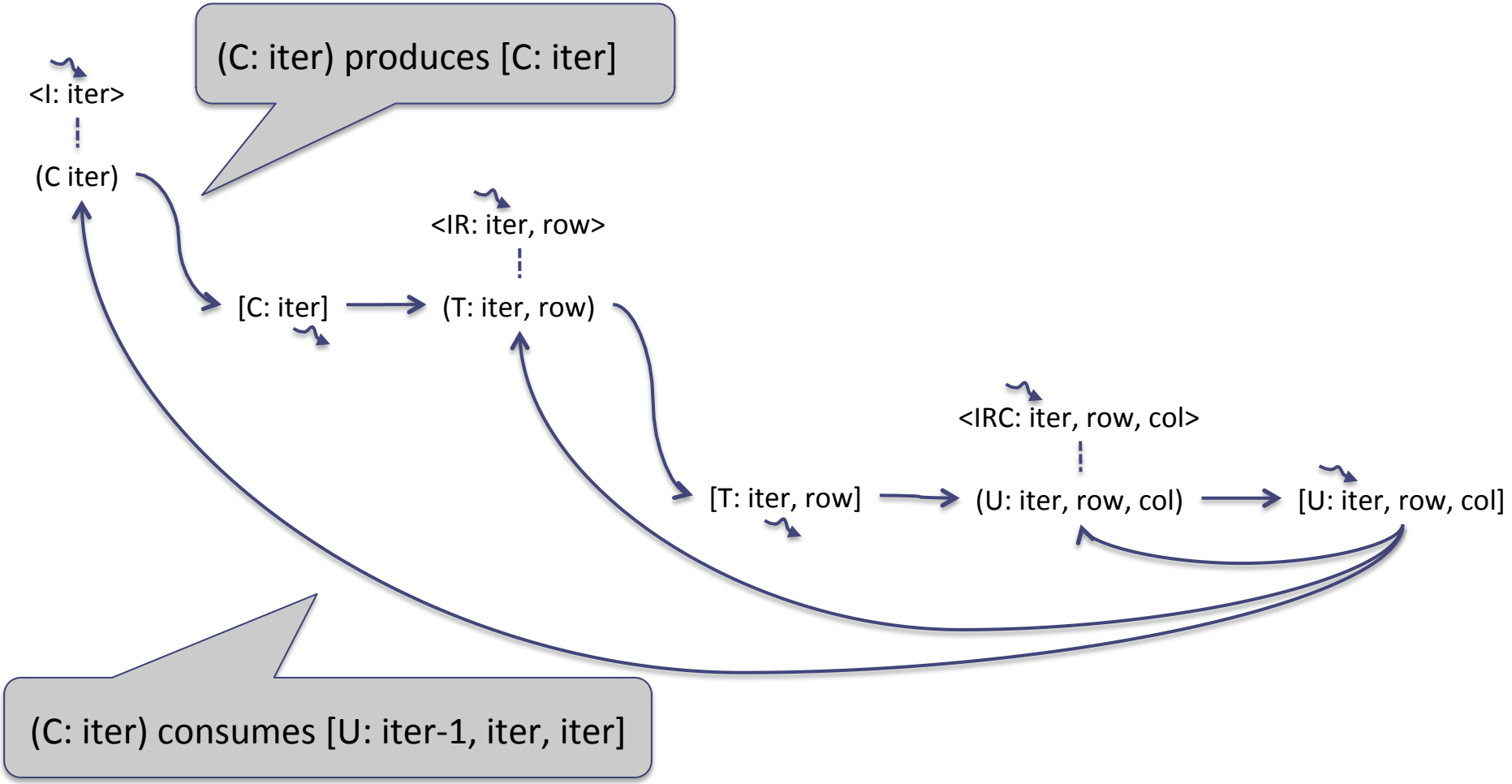
Cholesky CnC domain spec
4: exact set of instances (control)



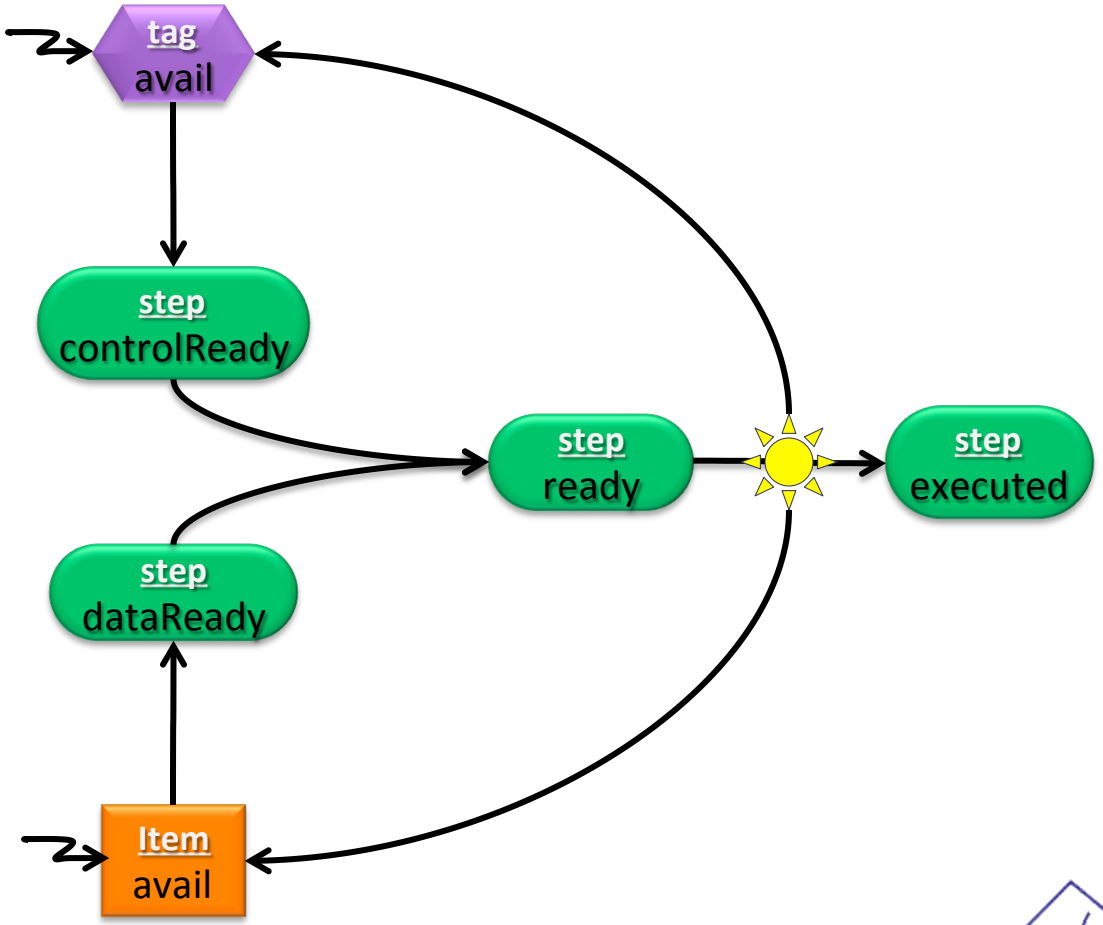
Cholesky CnC domain spec
4: exact set of instances (control)



Cholesky CnC domain spec
Optional: dependences functions



Semantics of CnC:
specifies a partial order of execution



Outline

1. Motivation
2. CnC
3. Wide array of existing CnC tuning approaches

Tuning

Tuning starts with the domain spec

- Just the required orderings

Wide range of tuning approaches

- Existing static tunings
- Existing dynamic tunings
- Possible future tunings

Static tuning

Vivek Sarkar

- Totally static schedule across time and space — Carl Offner, Alex Nelson
- Automatic conversion of element-level data and computation to tiled versions — Alina Sbirlea
- Polyhedral tiling — Alina Sbirlea, Jun Shirako
- Distribution functions across nodes in a cluster (dynamic within each node) — Frank Schlimbach
  - For data items or
  - For computation steps
- Minimize RT overhead by eliminating redundant attribute propagation — Zoran Budimlić, Kath Knobe, Tiago Cogumbreiro
- PIPES for distributed apps (Static and dynamic) — Martin Cong, Yuhan Peng

Dynamic tuning

- Runtimes based on OCR, TBB, Babel and Haskell — Vivek Sarkar
  - Nick Vrvilo, Zoran Budimlić, Vivek Sarkar
  - Frank Schlimbach
  - Shams Imam
  - Ryan Newton
- Basic: determines when a step is ready — Rice team, Frank Schlimbach
  - Tracks state changes and executes READY steps
- Workflow coordination — Frank Schlimbach, Yves Vandriessche
- Choice among CPU, GPU, FPGA — Alina Sbirlea, Zoran Budimlić
  - Dynamic compromise:

    Static step preference and dynamic device availability
- Hierarchical affinity groups for locality — Zoran Budimlić, Mike Burke, Sanjay Chatterjee, Kath Knobe, Nick Vrvilo
- Memory usage:

  Dynamic single assignment is higher level:

  Identifies values, not places

  Mapping data values to memory is tuning
  - Garbage collection: Get-count — Frank Schlimbach
  - Inspector/executor — Dragos Sbirlea

Possible future tunings

- Out-of-core computations
  - Based on hierarchical affinity groups
- Checkpoint-continue
  - Automatic continuous asynchronous checkpointing
- Collaborative runtimes
  - under development [Zoran Budimlić]
- Inspector/executor
  - For computation
- Demand-driven execution [Kath Knobe]
- Speculative execution

Isolation: Tuning from domain spec

- Critical for software engineering for exascale
- Critical for ease of tuning
- Just data and control dependences
  - No arbitrary constraints
  - Except for grain …
- CnC domain spec is totally flexible wrt tuning:
  - More approaches in mind now
  - New architectures
  - New tuning goals
  - New tuning approaches

Tuning grain size: Hierarchy

Kath Knobe
Nick Vrvilo

- Each computation step and data item can be decomposed
- Hierarchy adds more tuning flexibility
  - Choose the best hierarchy
  - Choose the best grain
  - Different tunings can be applied at different levels in the hierarchy

Conclusion

- CnC is a dependence language
  - Control and data dependences
  - Explicit, distinct and at the same level
- Tuning is separate
  - No CnC-specific tuning
- This isolation makes tuning easier and more flexible

Thanks to …

DARPA: UHPC

DOE: X-Stack

Cambridge Research Lab

DEC / Compaq / HP / Intel

Carl Offner, Alex Nelson

Intel

Frank Schlimbach, James Brodman, Mark Hampton, Geoff Lowney, Vincent Cave, Kamal Sharma, X-Stack TG team

Rice

Vivek Sarkar, Zoran Budimlić, Mike Burke, Sanjay Chatterjee, Nick Vrvilo, Martin Kong, Tiago Cogumbreiro

Reservoir

Rich Lethin

Benoit Meister

UC Irvine

Aparna Chandramowlishwaran (Intel intern)

Google

Alina Sbirlea (Rice)

Dragos Sbirlea (Rice)

UCSD

Laura Carrington

Pietro Cicotti

GaTech

Rich Vuduc

Hasnain Mandviwala (Inel intern)

Kishore Ramachandran

Indiana

Ryan Newton (Intel)

Purdue

Milind Kulkarni

Chenyang Liu

PNNL

John Feo

Ellen Porter

Facebook

Nicolas Vasilache (Reservoir)

Micron

Kyle Wheeler (xxNL)

Dreamworks

Martin Watt

Two Sigma

Shams Imam (Rice)

Sagnak Tasirlar (Rice

- Intel

  CnC on Intel's WhatIf site

  http://software.intel.com/en-us/articles/intel-concurrent-collections-for-cc

  Available open source

  https://icnc.github.io

- Rice

  https://wiki.rice.edu/confluence/display/HABANERO/CNC

- Discuss CnC related topics

  New apps, optimizations, runtime, tuning, … Send mail to

  kath.knobe@rice.edu, zoran@rice.edu, or frank.schlimbach@intel.com

- CnC'16 workshop

  Sept 27-28, 2016 Co-located with LCPC'16 in Rochester, NY

  https://cncworkshop2016.github.io/

- To get on mailing list send mail to

  kath.knobe@rice.edu, zoran@rice.edu, or frank.schlimbach@intel.com

Thank you!