Topic 14: Parallelism

COS 320

Compiling Techniques

Princeton University
Spring 2018

Prof. David August

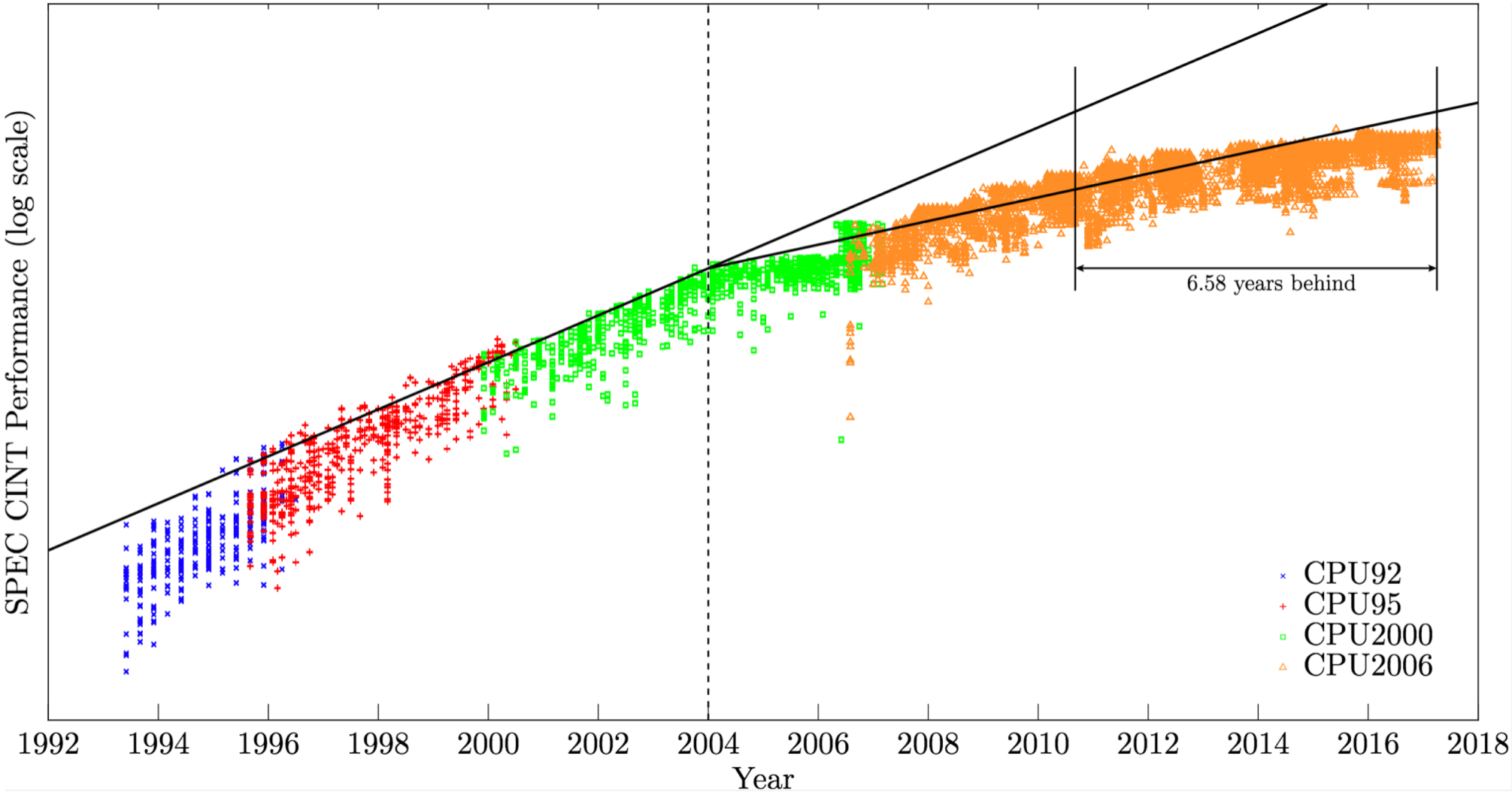
Final Exam!

- Thursday May 3 in class
- Closed book, closed notes

Moore's Law

Source: Intel/Wikipedia

Single-Threaded Performance Not Improving

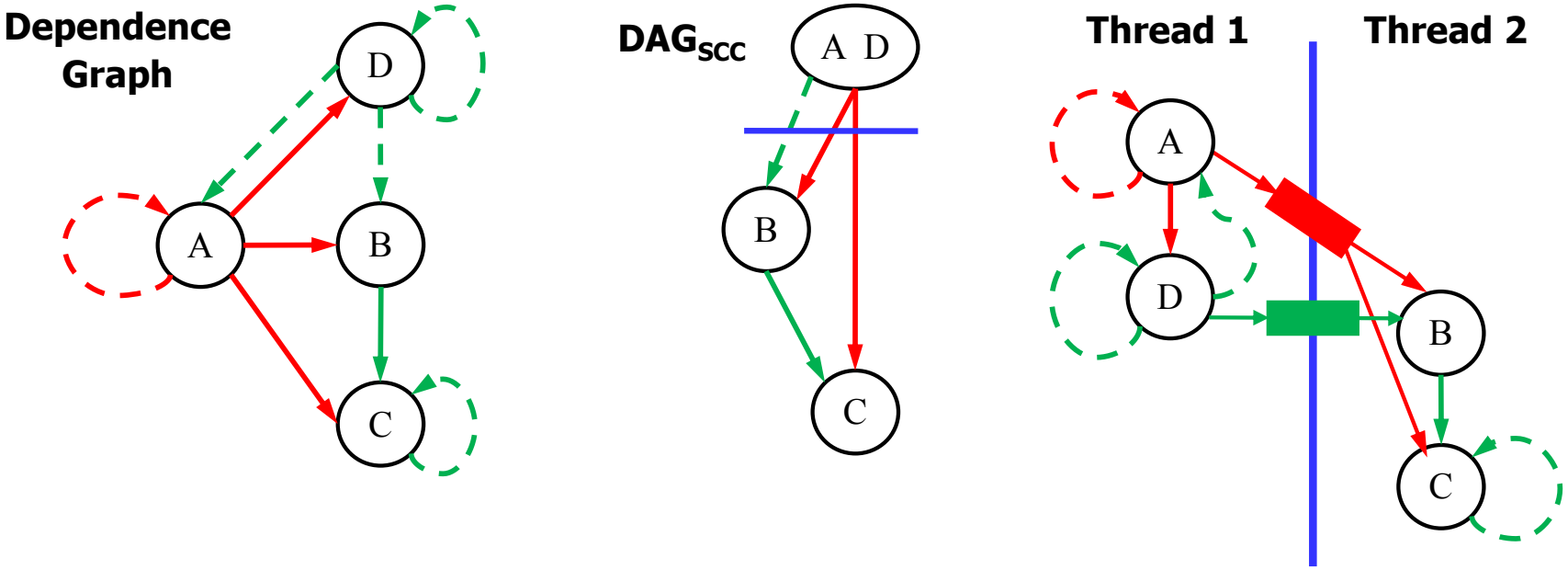


Decoupled Software Pipelining

[MICRO 2005]

Decoupled Software Pipelining (DSWP)

```
A: while(node)
B:   ncost = doit(node);
C:   cost += ncost;
D:   node = node->next;
```



register

control

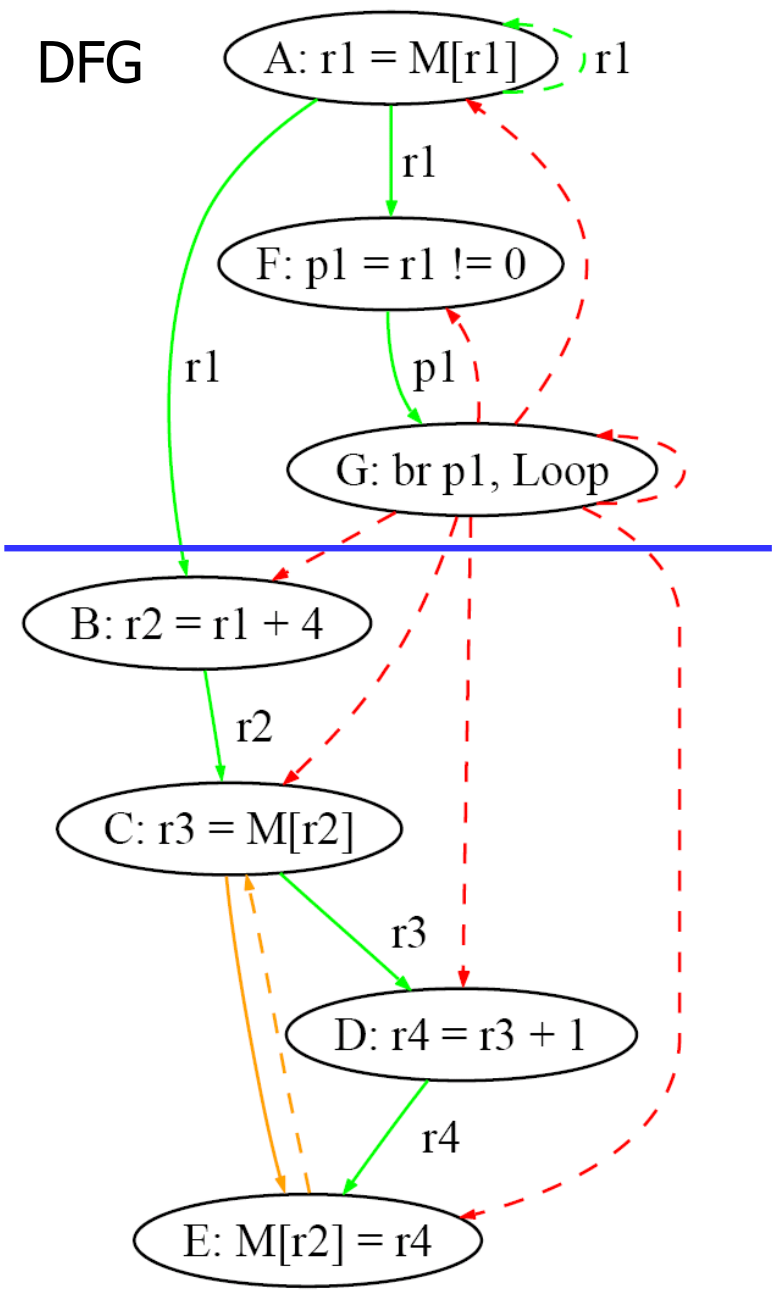
intra-iteration

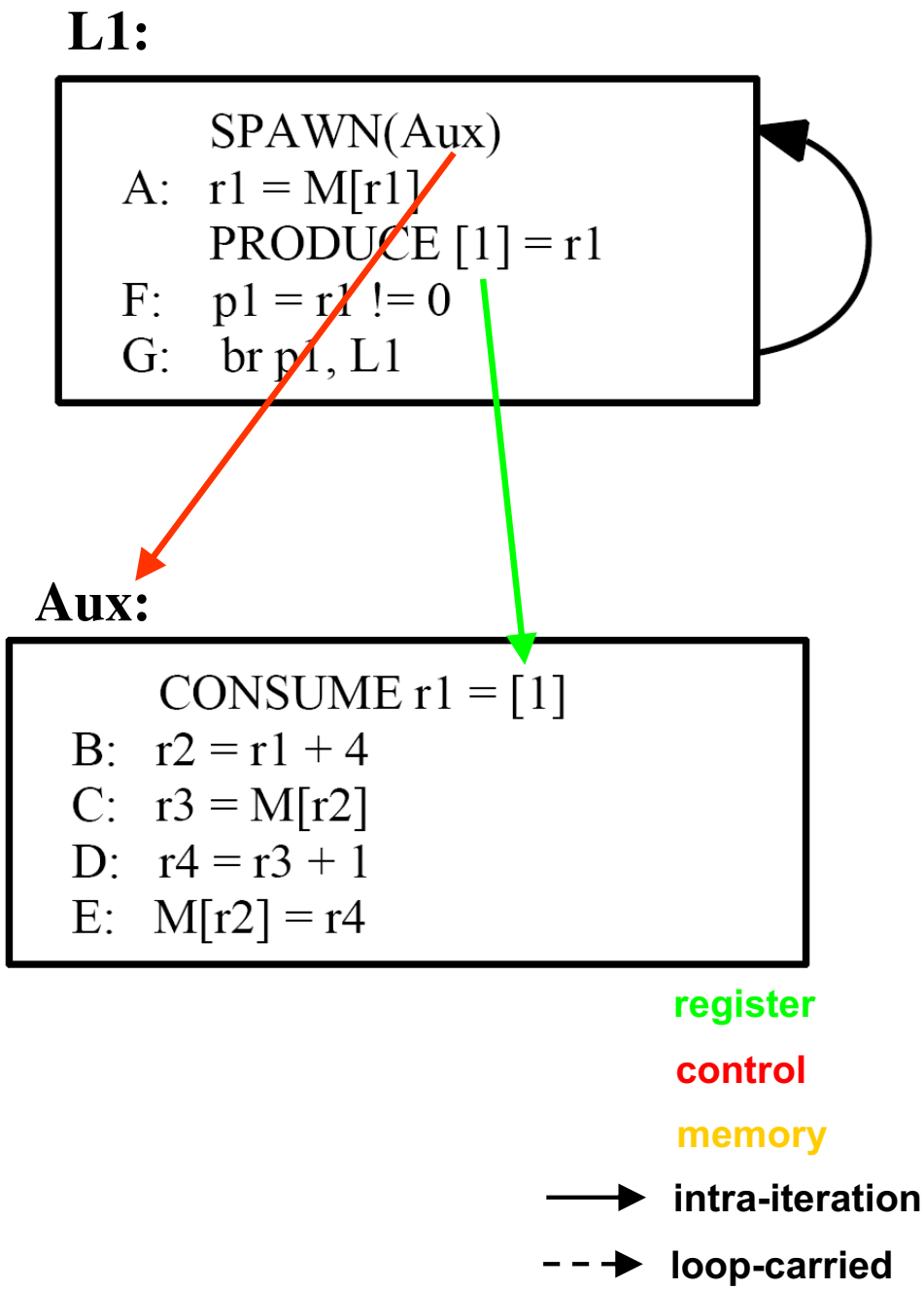
loop-carried

communication queue

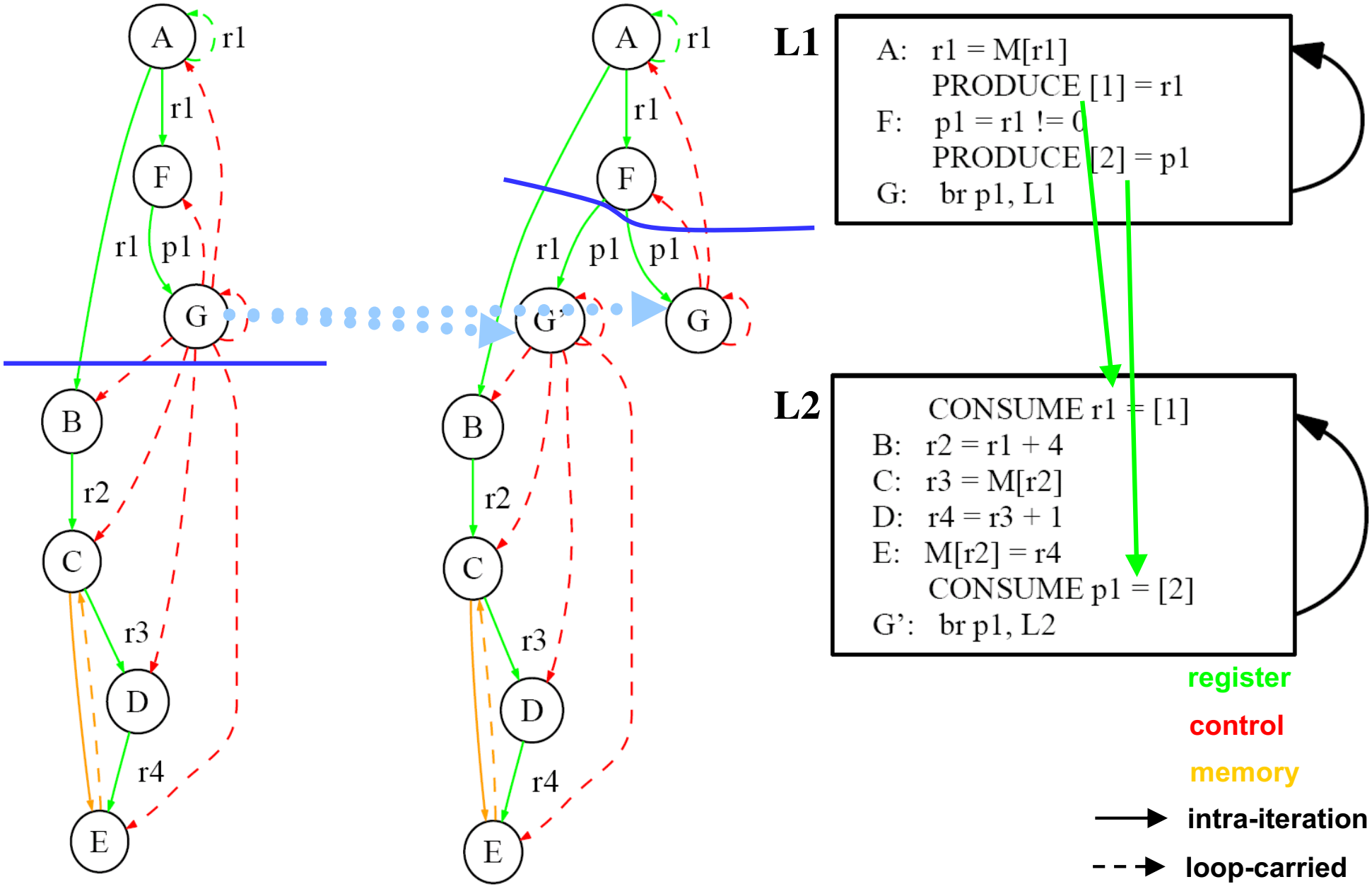
Inter-thread communication latency is a one-time cost

Implementing DSWP

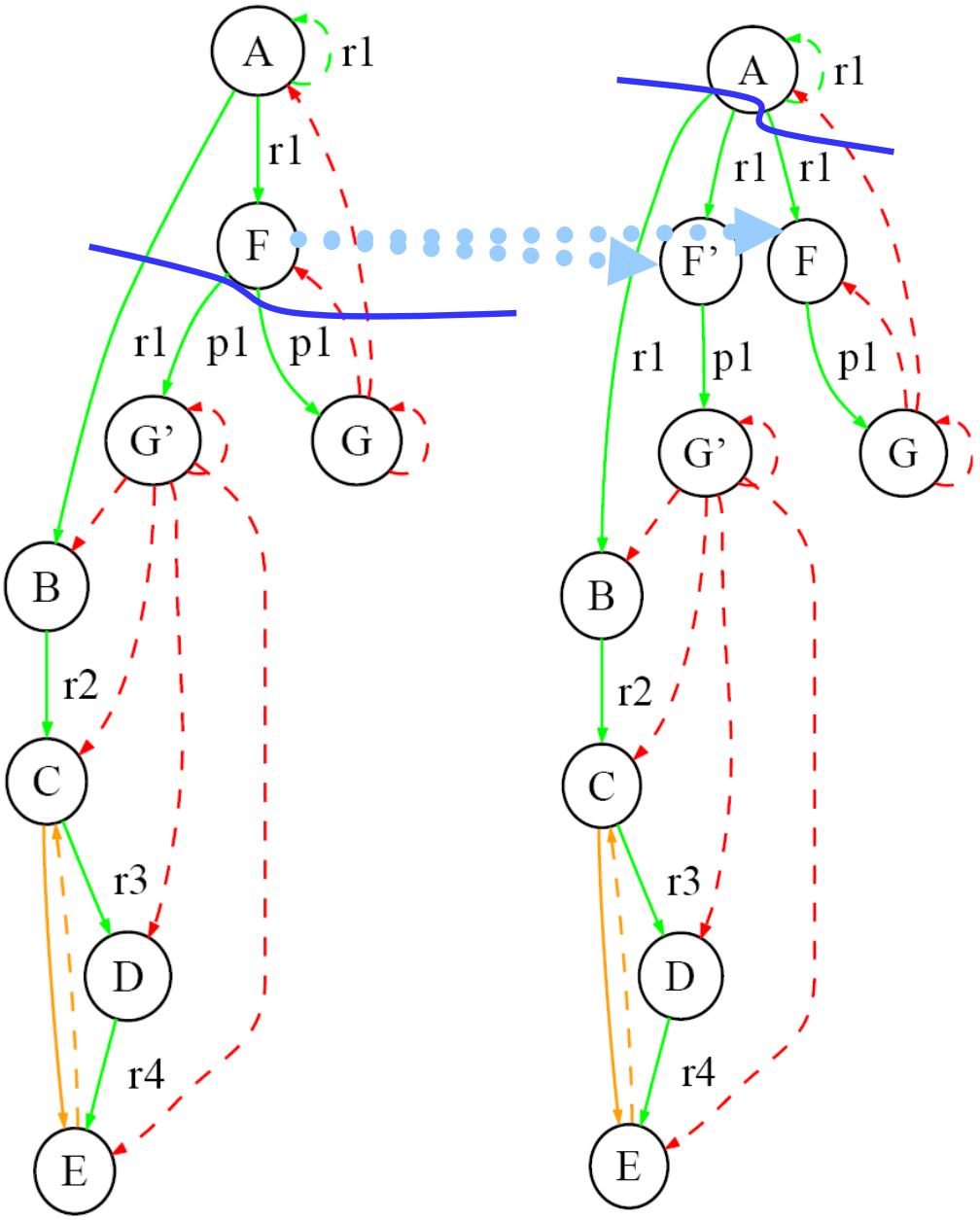




Optimization: Node Splitting To Eliminate Cross Thread Control



Optimization: Node Splitting To Reduce Communication



L1

```
A:  r1 = M[r1]
    PRODUCE [1] = r1
F:  p1 = r1 != 0
G:  br p1, L1
```

L2

```
    CONSUME r1 = [1]
B:  r2 = r1 + 4
C:  r3 = M[r2]
D:  r4 = r3 + 1
E:  M[r2] = r4
F': p1 = r1 != 0
G': br p1, L2
```

register

control

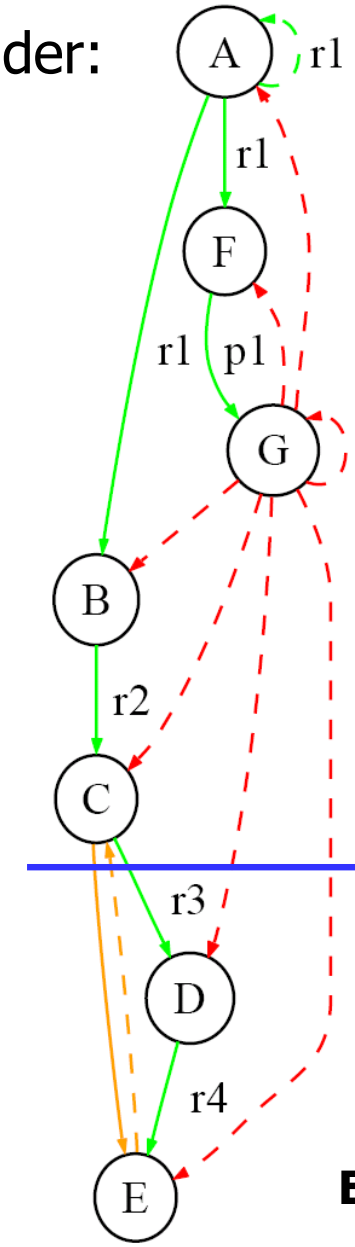
memory

intra-iteration

loop-carried

Constraint: Strongly Connected Components

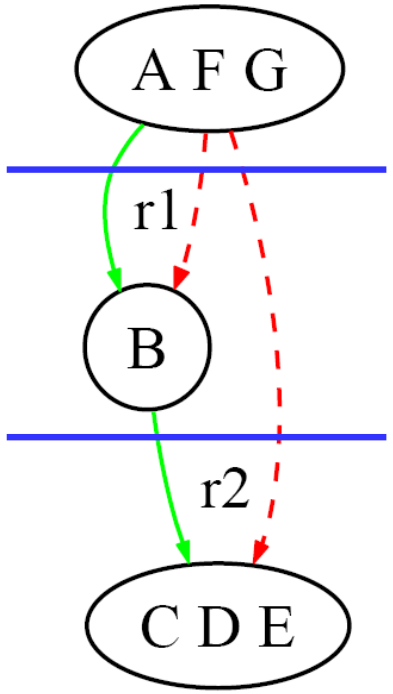
Consider:




```
      SPAWN(Aux)
A:    r1 = M[r1]
B:    r2 = r1 + 4
C:    r3 = M[r2]
      PRODUCE [1] = r3
      CONSUME r0 = [2]
F:    p1 = r1 != 0
G:    br p1, L1
```

```
      CONSUME r3 = [1]
D:    r4 = r3 + 1
E:    M[r2] = r4
      PRODUCE [2] = r0
```

Solution: DAG_{SCC}



Eliminates pipelined/decoupled property

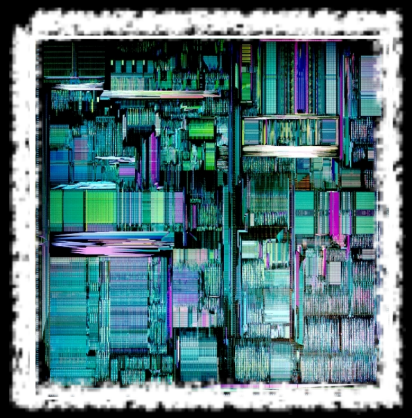
register

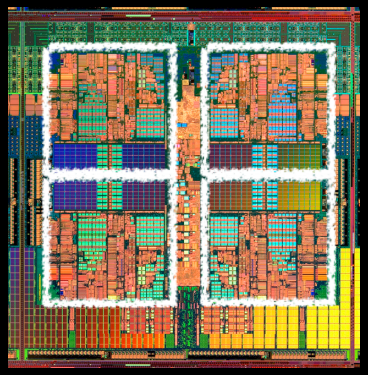
control

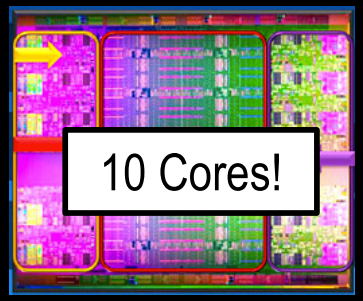
memory

intra-iteration

loop-carried



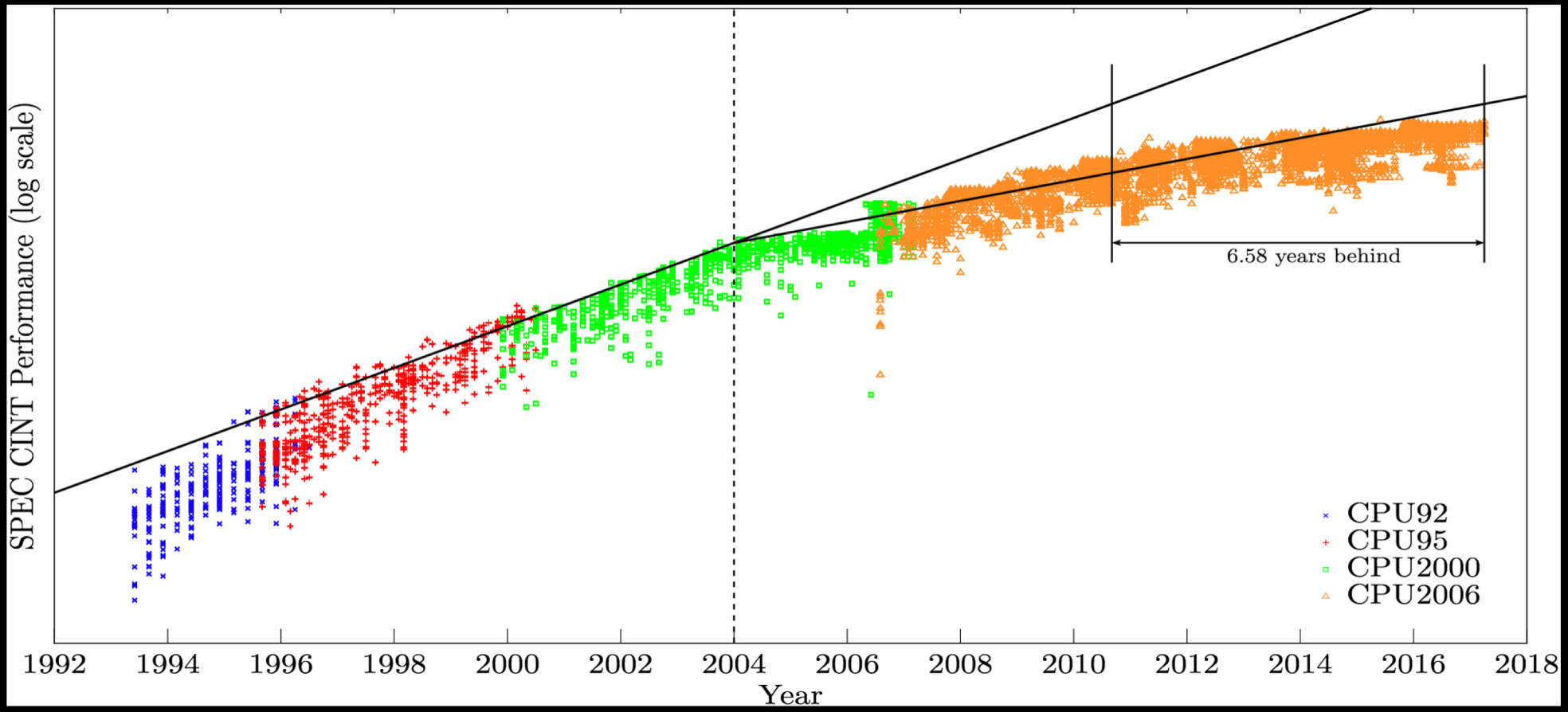




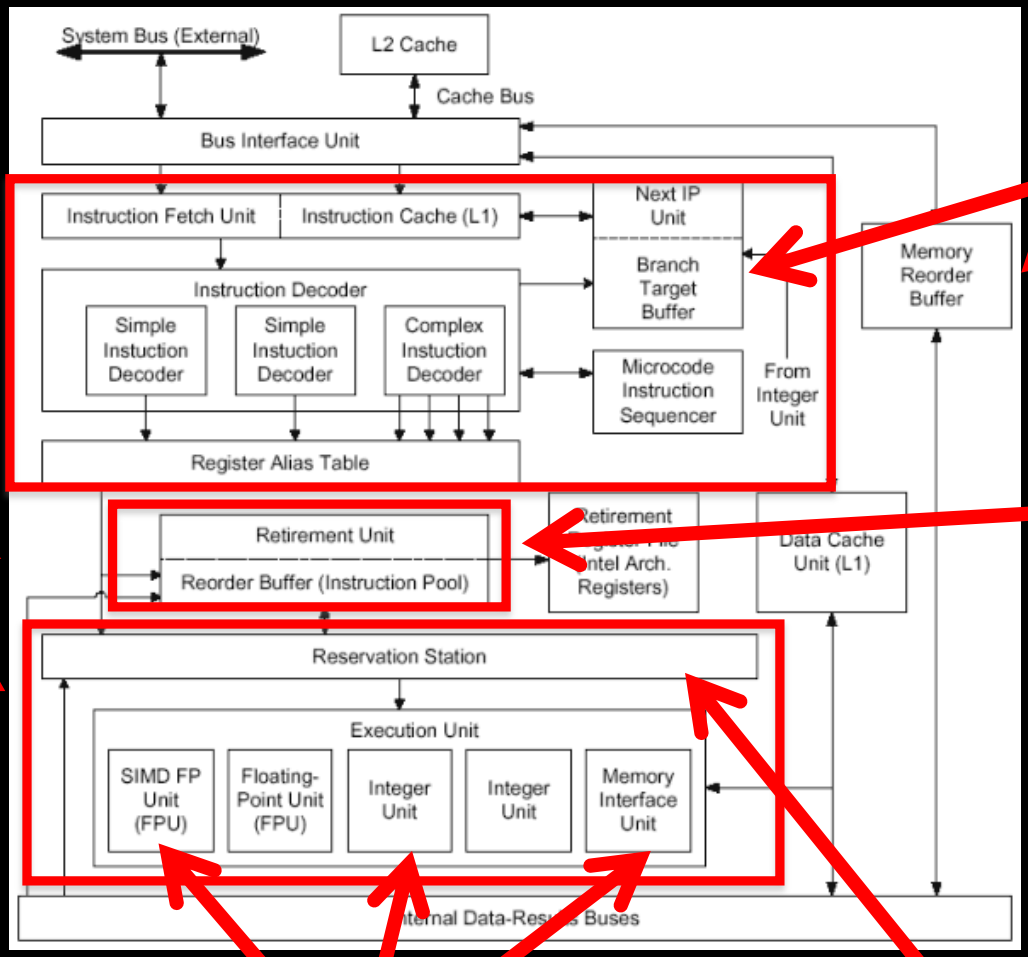
10-Core Intel Xeon
"Unparalleled Performance"

Era of DIY:

- Multicore
- Reconfigurable
- GPUs
- Clusters



P6 SUPERSCALAR ARCHITECTURE (CIRCA 1994)



Automatic Pipelining

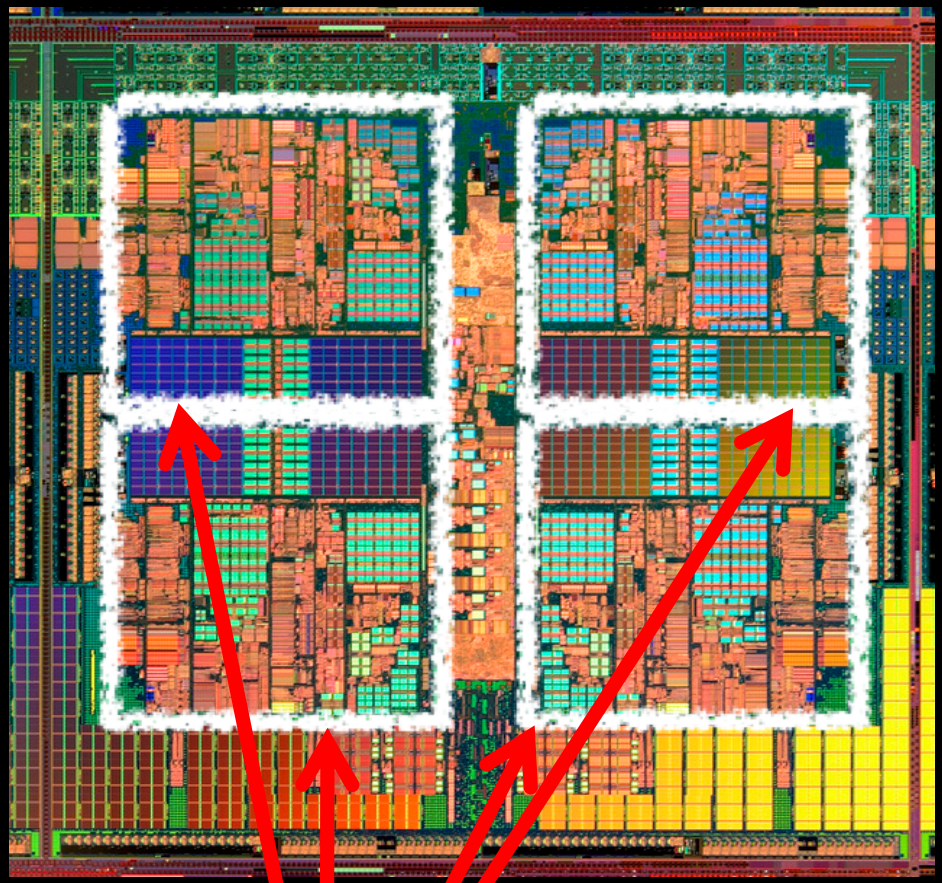
Automatic Speculation

Commit

Parallel Resources

Automatic Allocation/Scheduling

Multicore Architecture (Circa 2010)



~~Automatic Pipelining~~

~~Automatic Speculation~~

~~Commit~~

Parallel Resources

~~Automatic Allocation/Scheduling~~

ABCPL
ACE
ACT++
Active messages
Adl
Adsmith
ADDAP
AFAPI
ALWAN
AM
AMDC
AppLeS
Amoeba
ARTS
Athapascan-0b
Aurora
Automap
bb_threads
Blaze
BSP
BlockComm
C*.
"C* in C
C**
CarlOS
Cashmere
C4
CC++
Chu
Charlotte
Charm
Charm++
Cid
Cilk
CM-Fortran
Converse
Code
COOL

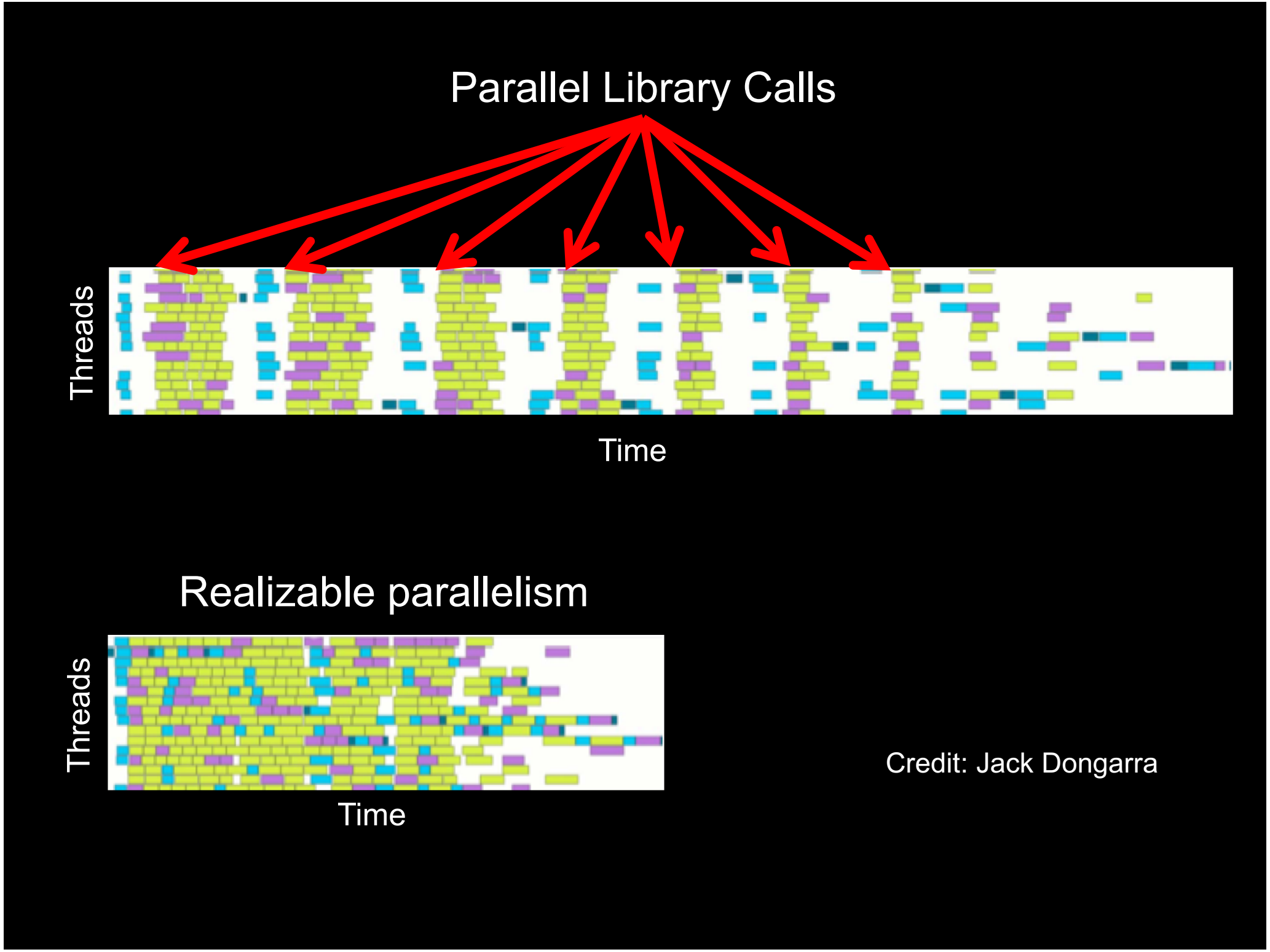
CORRELATE
CPS
CRL
CSP
Cthreads
CUMULVS
DAGGER
DAPPLE
Data Parallel C
DC++
DCE++
DDD
DICE.
DIPC
DOLIB
DOME
DOSMOS.
DRL
DSM-Threads
Ease .
ECO
Eiffel
Eilean
Emerald
EPL
Excalibur
Express
Falcon
Filaments
FM
FLASH
The FORCE
Fork
Fortran-M
FX
GA
GAMMA
Glenda

GLU
GUARD
HAsL.
Haskell
HPC++
JAVAR.
HORUS
HPC
IMPACT
ISIS.
JAVAR
JADE
Java RMI
javaPG
JavaSpace
JIDL
Joyce
Khoros
Karma
KOAN/Fortran-S
LAM
Lilac
Linda
JADA
WWWinda
ISETL-Linda
ParLin
Eilean
P4-Linda
Glenda
POSYBL
Objective-Linda
LiPS
Locust
Lparx
Lucid
Maisie
Manifold

Mentat
Legion
Meta Chaos
Midway
Millipede
CparPar
Mirage
MpC
MOSIX
Modula-P
Modula-2*
Multipol
MPI
MPC++
Munin
Nano-Threads
NESL
NetClasses++
Nexus
Nimrod
NOW
Objective Linda
Occam
Omega
OpenMP
Orca
OOF90
P++
P3L
p4-Linda
Pablo
PADE
PADRE
Panda
Papers
AFAPI.
Para++
Paradigm

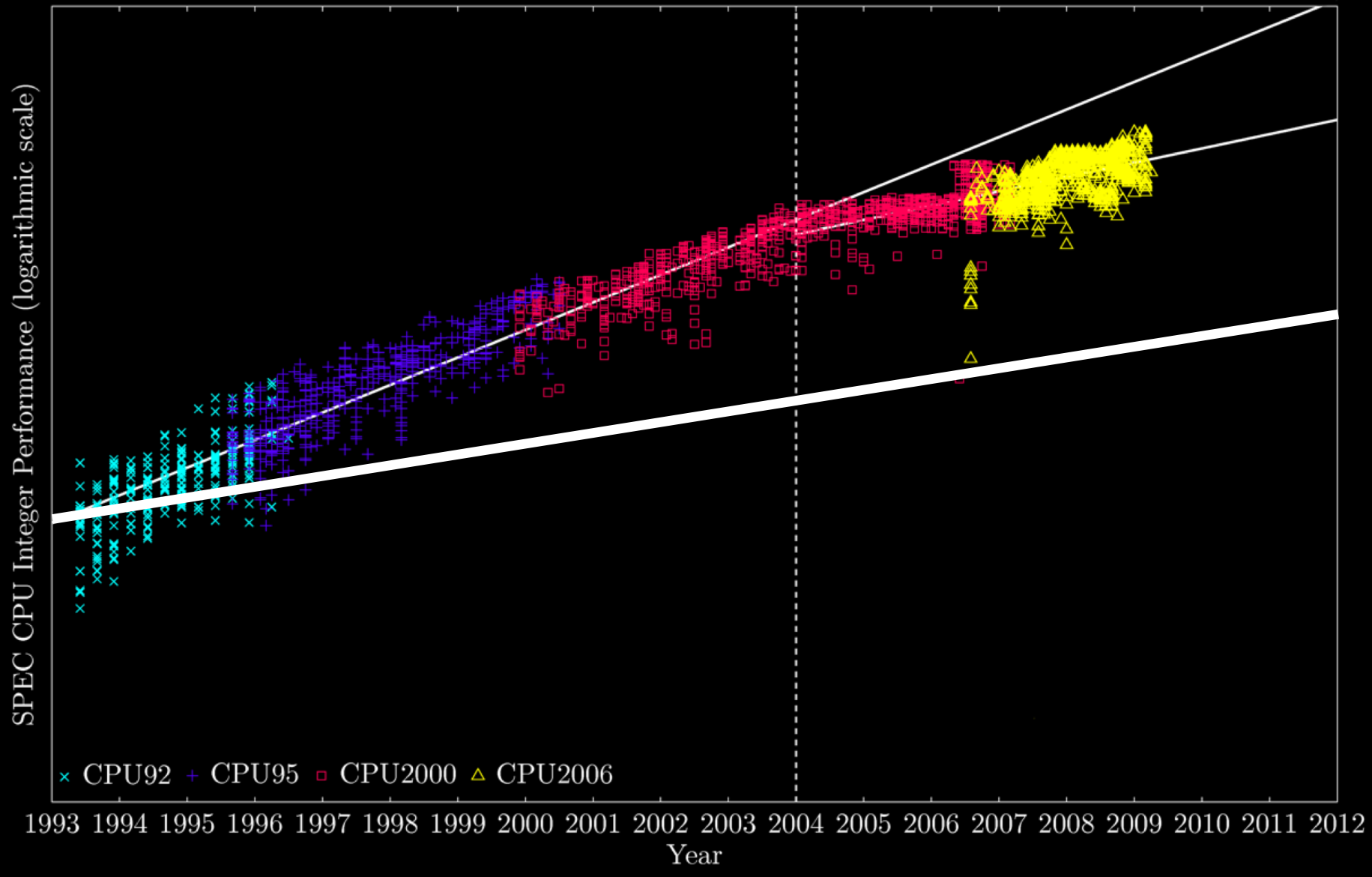
Parafrase2
Paralation
Parallel-C++
Parallaxis
ParC
ParLib++
ParLin
Parmacs
Parti
pC
pC++
PCN
PCP:
PH
PEACE
PCU
PET
PETSc
PENNY
Phosphorus
POET.
Polaris
POOMA
POOL-T
PRESTO
P-RIO
Prospero
Proteus
QPC++
PVM
PSI
PSDM
Quake
Quark
Quick Threads
Sage++
SCANDAL
SAM

pC++
SCHEDULE
SciTL
POET
SDDA.
SHMEM
SIMPLE
Sina
SISAL.
distributed smalltalk
SMI.
SONiC
Split-C.
SR
Sthreads
Strand.
SUIF.
Synergy
Telegrphos
SuperPascal
TCGMSG.
Threads.h++.
TreadMarks
TRAPPER
uC++
UNITY
UC
V
ViC*
Visifold V-NUS
VPE
Win32 threads
WinPar
WWWinda
XENOOPS
XPC
Zounds
ZPL

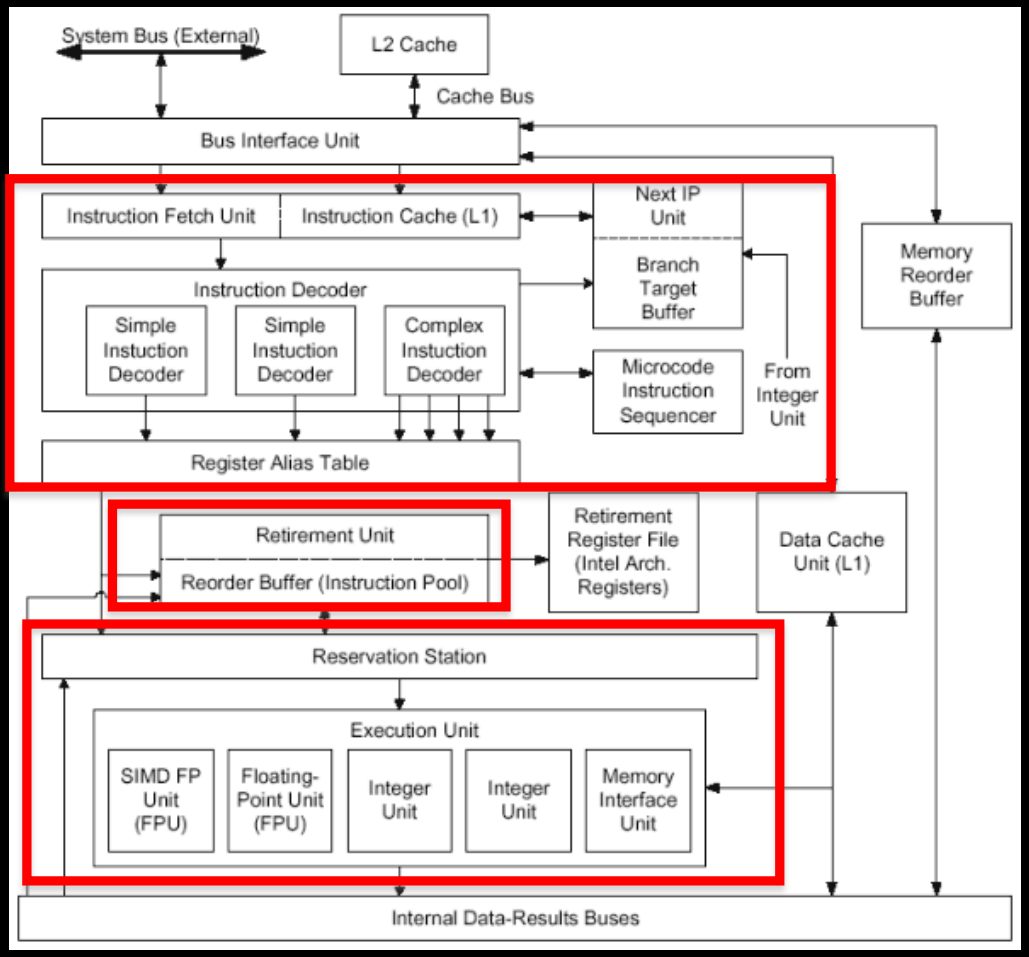

Parallel Library Calls
Threads
Time
Realizable parallelism
Threads
Time
Credit: Jack Dongarra

"Compiler Advances Double Computing Power Every 18 Years!"

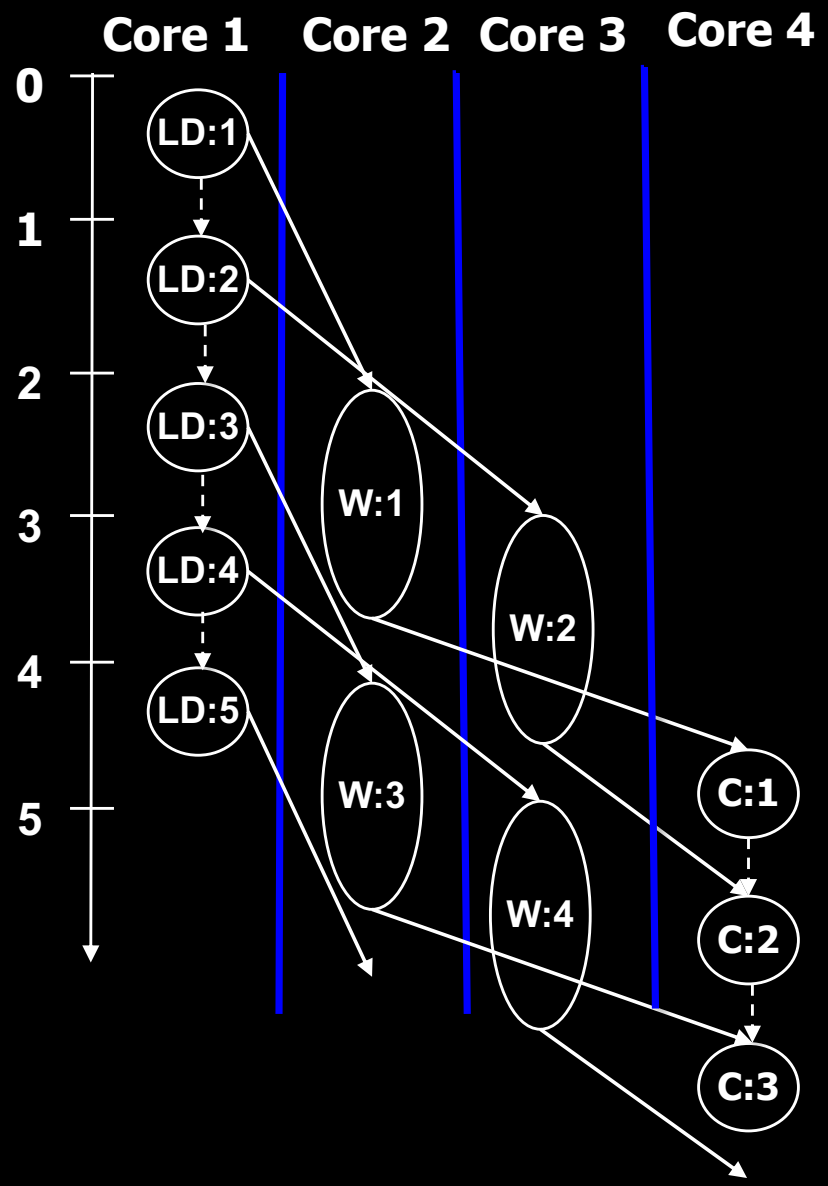
– Proebsting's Law



P6 SUPERSCALAR ARCHITECTURE



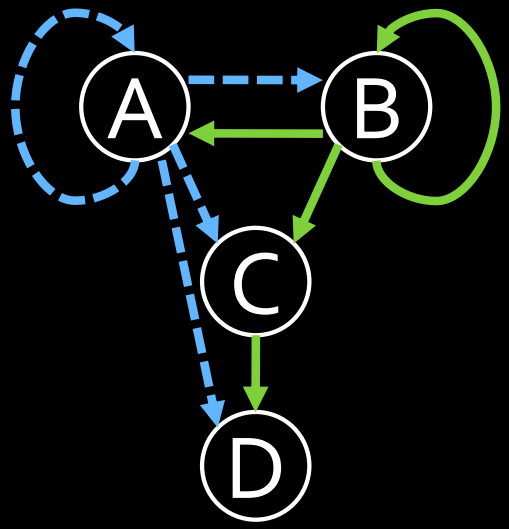
Spec-PS-DSWP



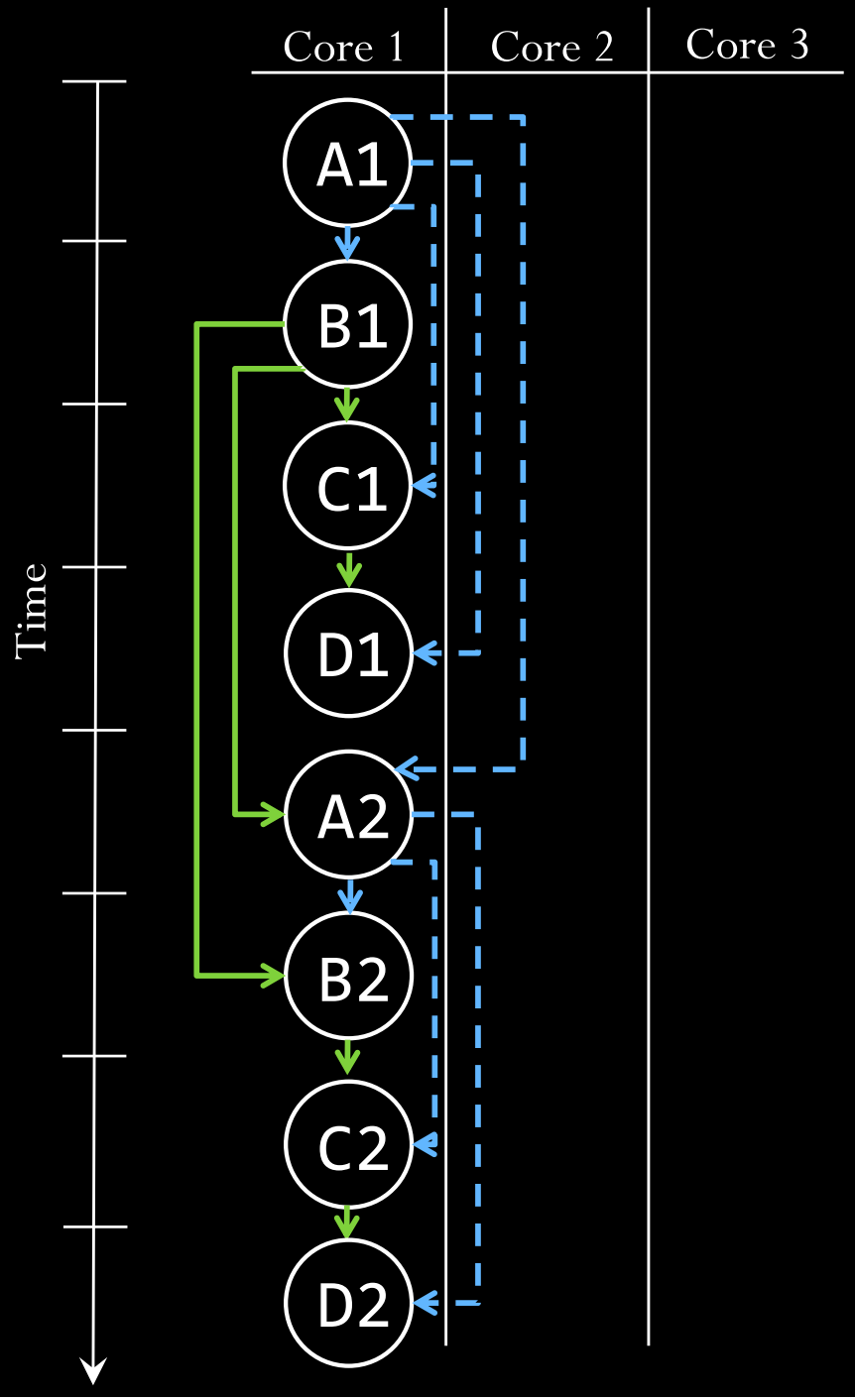
Example

```
A: while (node) {
B:    node = node->next;
C:    res = work(node);
D:    write(res);
   }
```

Program Dependence Graph



Control Dependence
Data Dependence

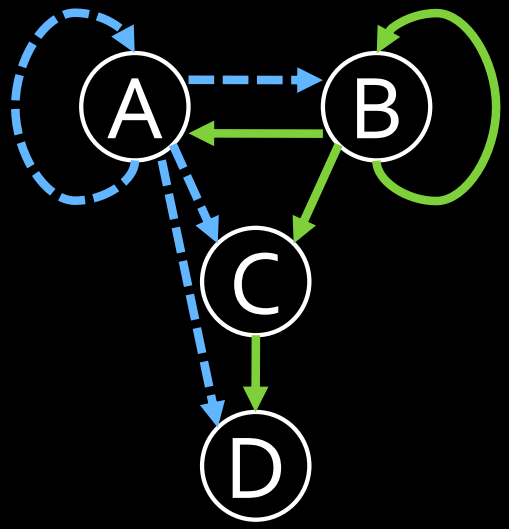


Spec-DOALL

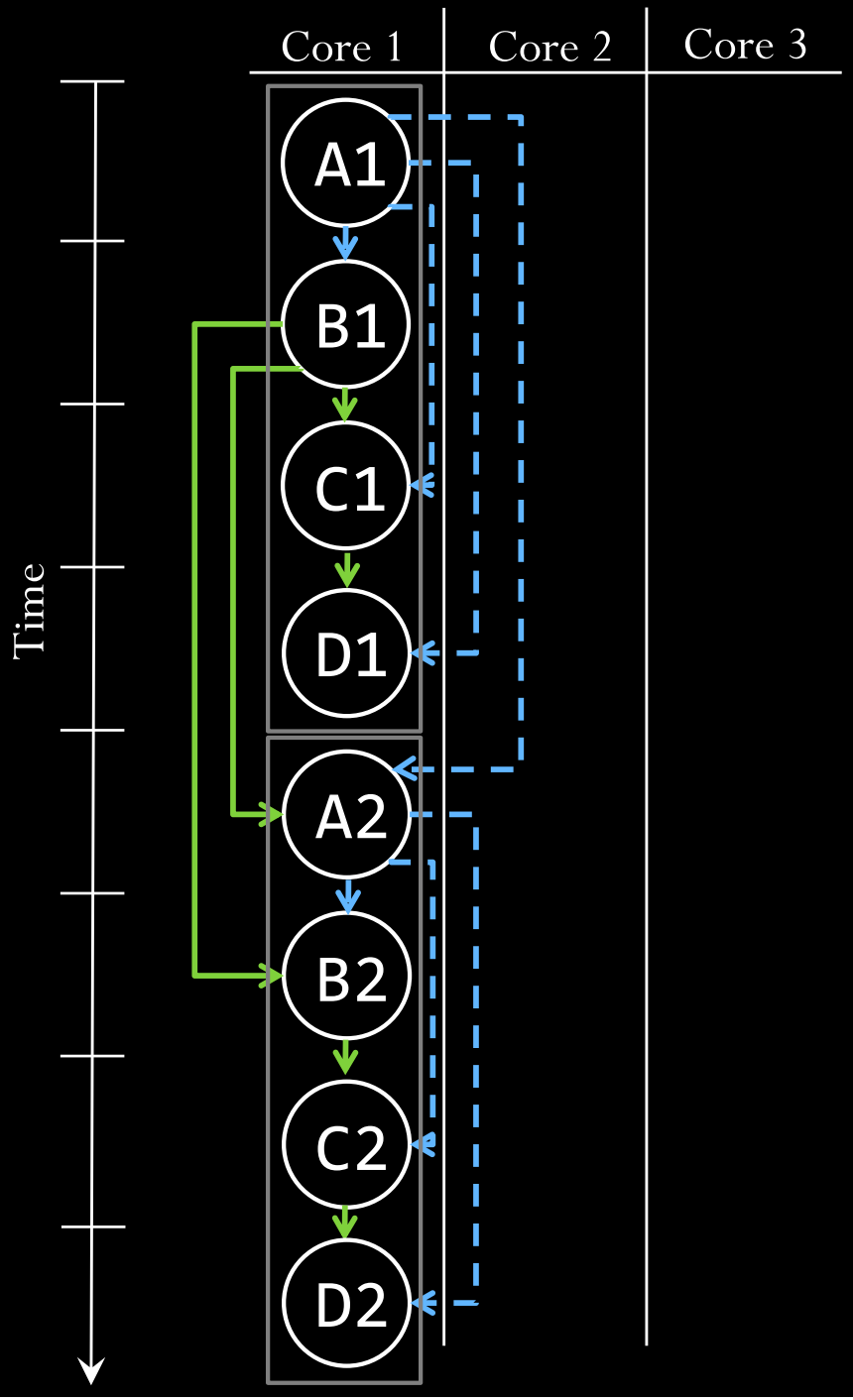
Example

```
A: while (node) {
B:    node = node->next;
C:    res = work(node);
D:    write(res);
   }
```

Program Dependence Graph



Control Dependence
Data Dependence

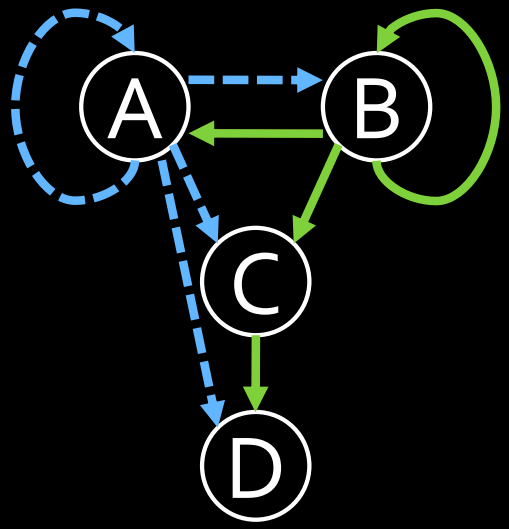


Spec-DOALL

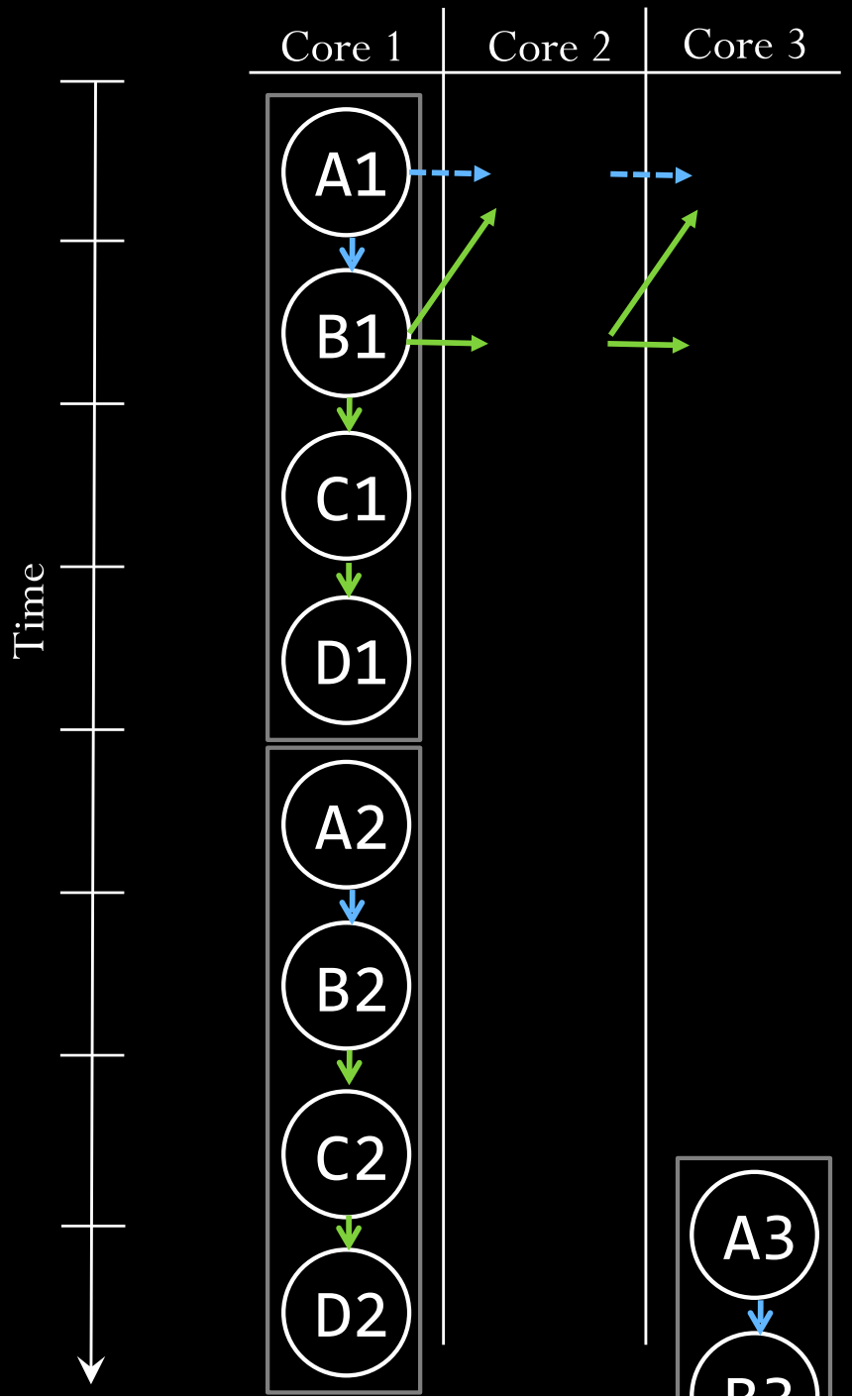
Example

```
A: while (node) {
B:    node = node->next;
C:    res = work(node);
D:    write(res);
   }
```

Program Dependence Graph



Control Dependence
Data Dependence

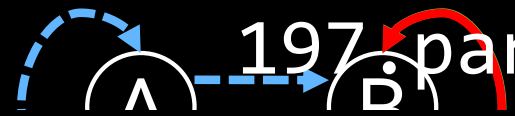


Spec-DOALL

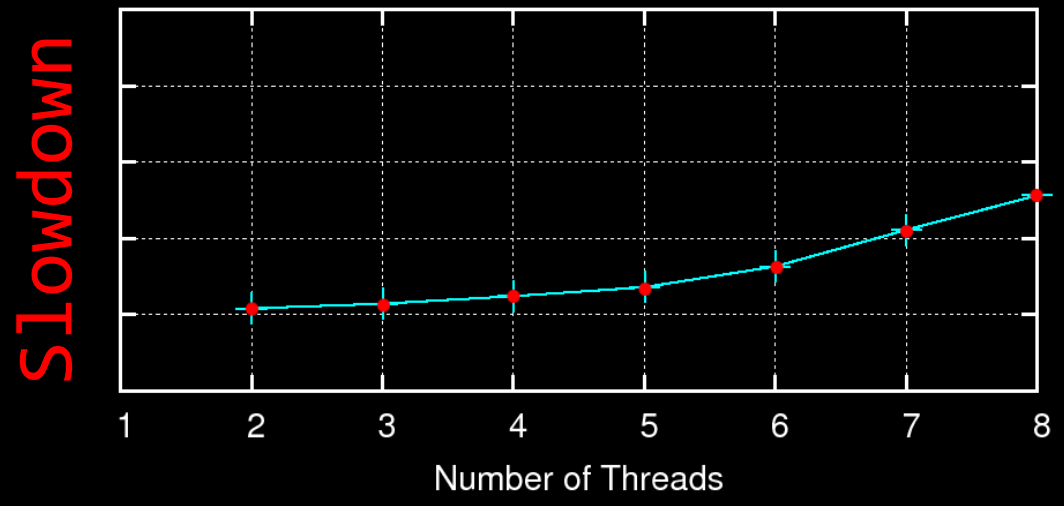
Example

```
A: while (node) {
B:    node = node->next;
C:    res = work(node);
D:    write(res);
   }
```

Program Dependence Graph

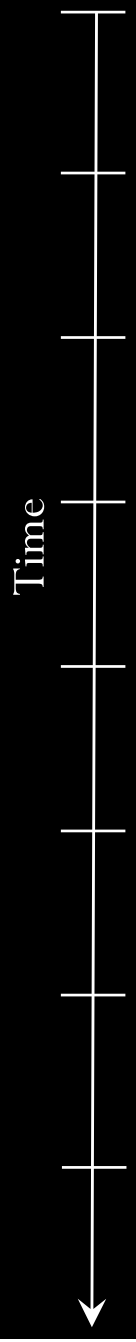


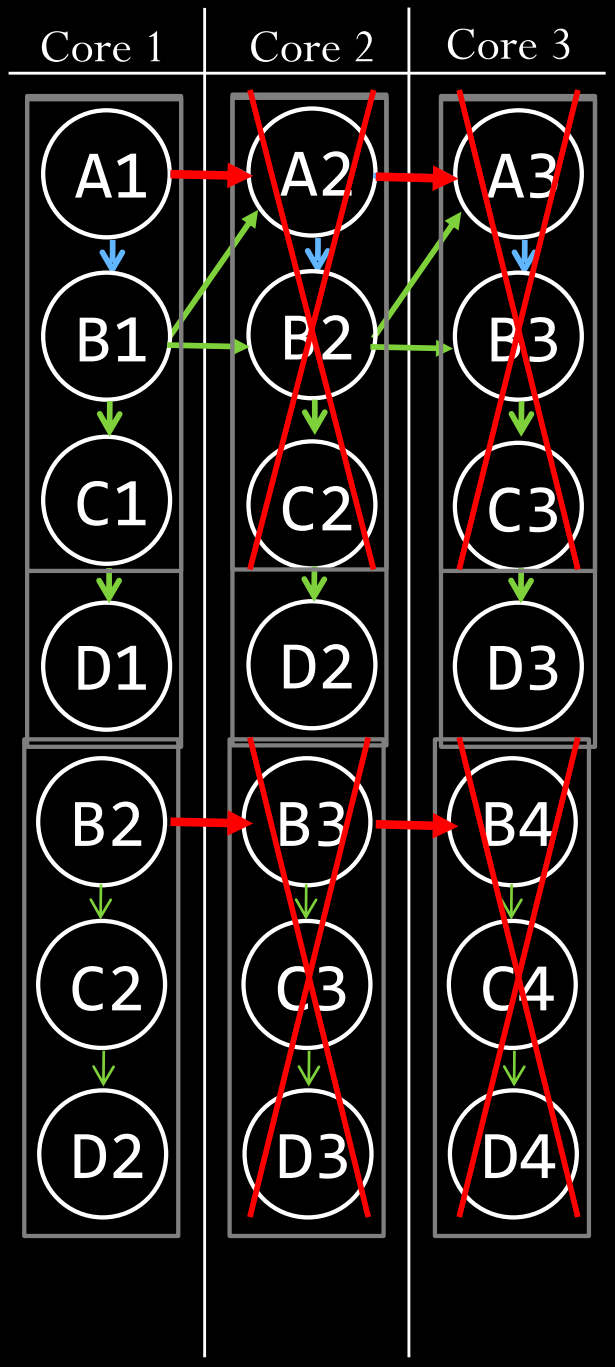
197.parser

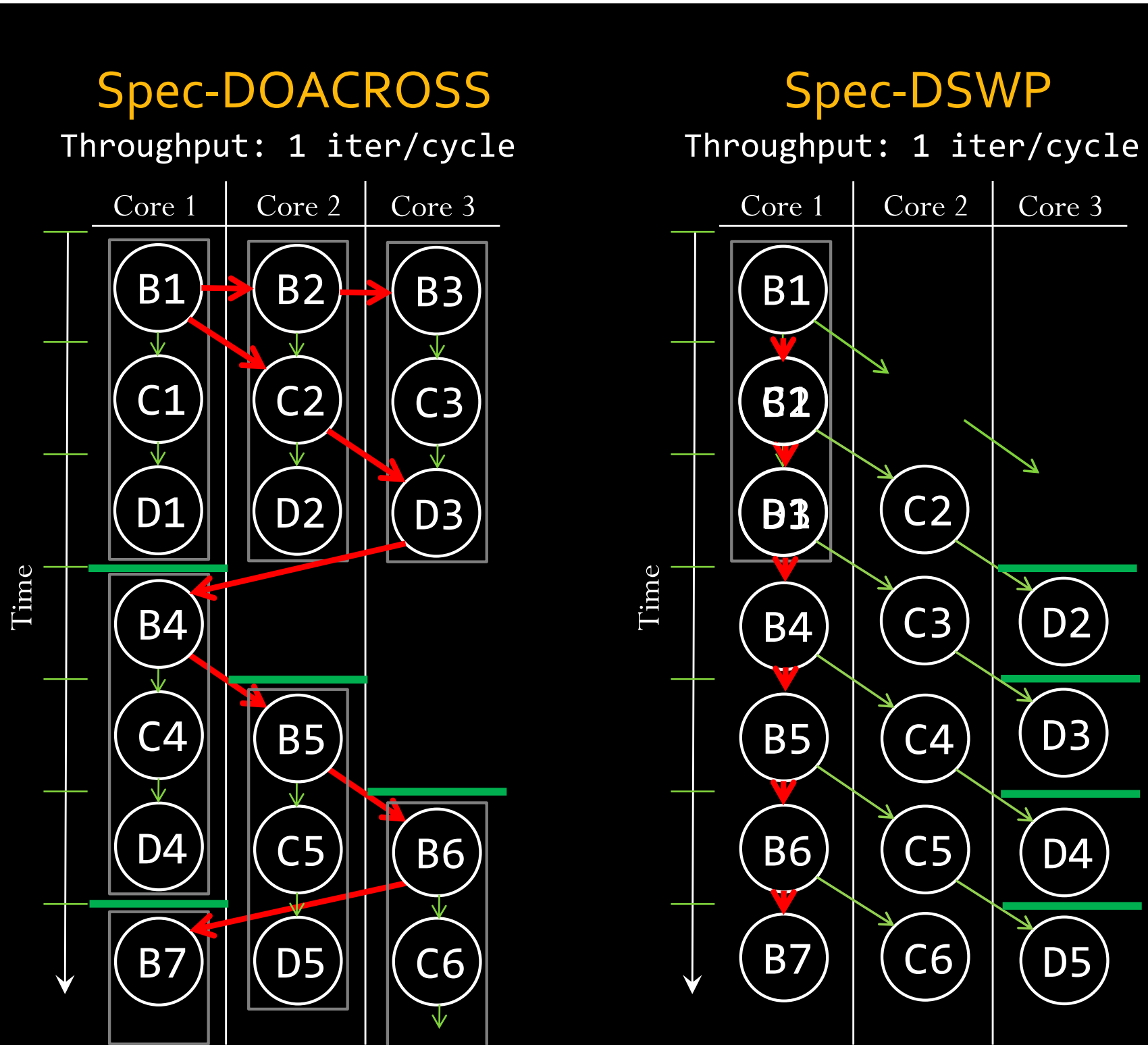


Control Dependence

Data Dependence

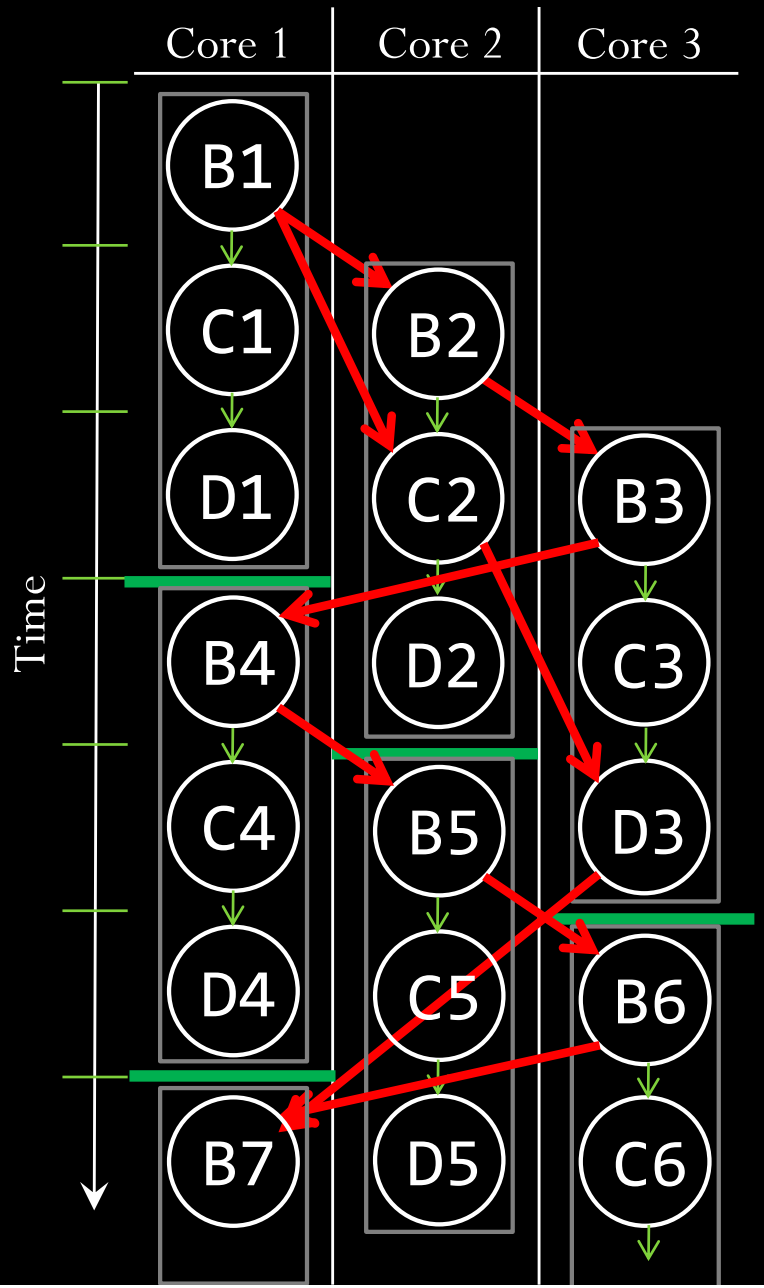




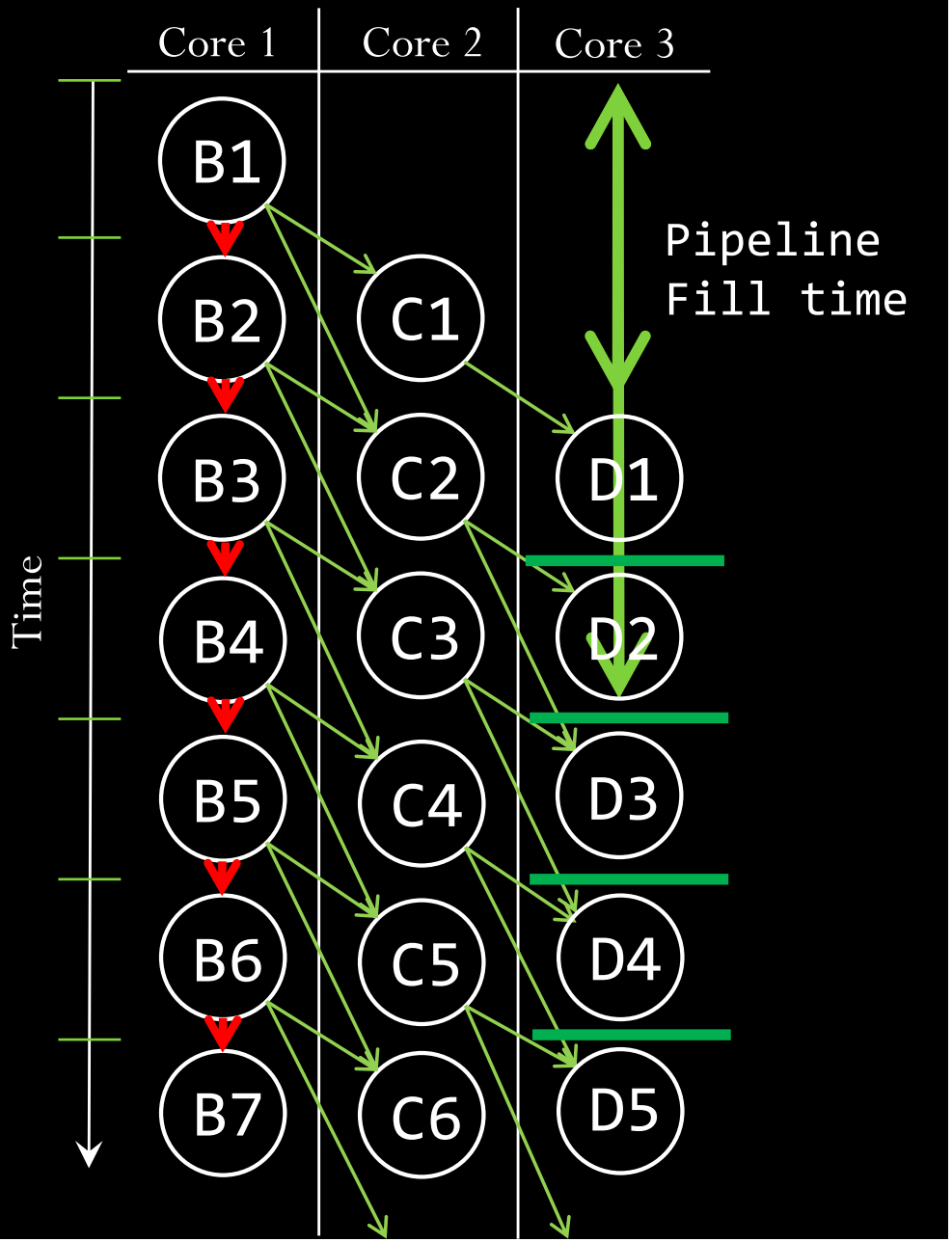

Spec-DOACROSS
Throughput: 1 iter/cycle
Core 1
Core 2
Core 3
Time
B1
B2
B3
C1
C2
C3
D1
D2
D3
B4
B5
B6
C4
C5
C6
D4
D5
B7
Spec-DSWP
Throughput: 1 iter/cycle
Core 1
Core 2
Core 3
Time
B1
B2
B3
C2
B4
C3
D2
B5
C4
D3
B6
C5
D4
B7
C6
D5

Comparison: Spec-DOACROSS and Spec-DSWP

Comm.Latency = 1: 1 iter/cycle
Comm.Latency = 2: **0.5 iter/cycle**



Comm.Latency = 1: 1 iter/cycle
Comm.Latency = 2: **1 iter/cycle**



Spec-DOACROSS vs. Spec-DSWP

[MICRO 2010]

Geomean of 11 benchmarks on the same cluster