Topic 14: Parallelism

COS 320

Compiling Techniques

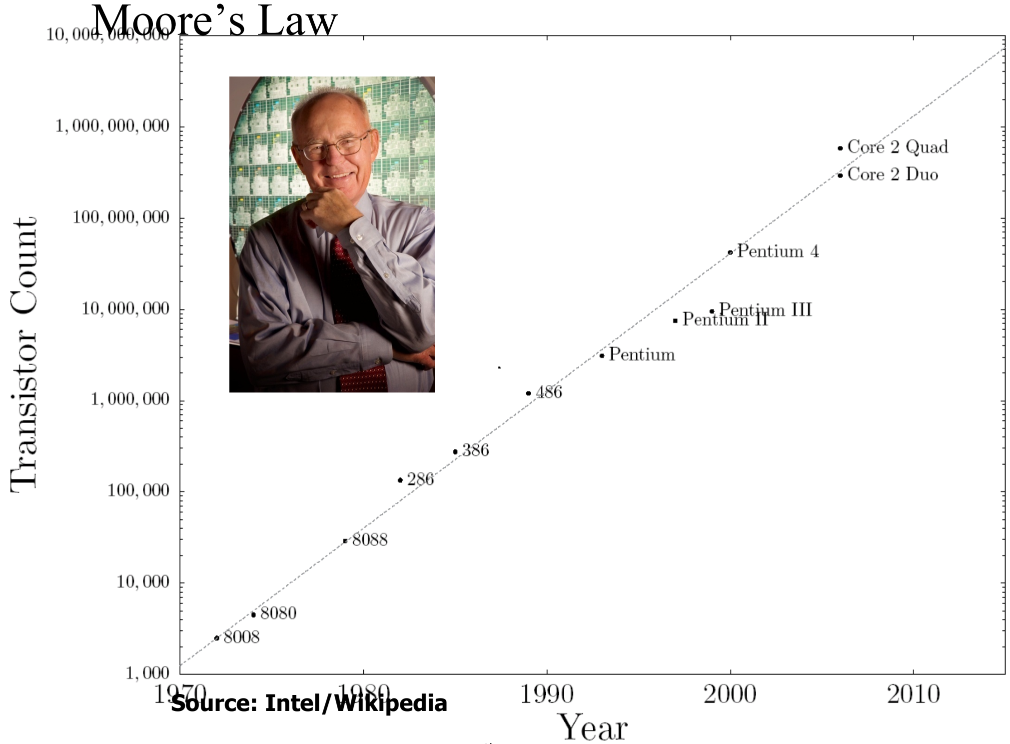
Princeton University
Spring 2018

Prof. David August

Final Exam!

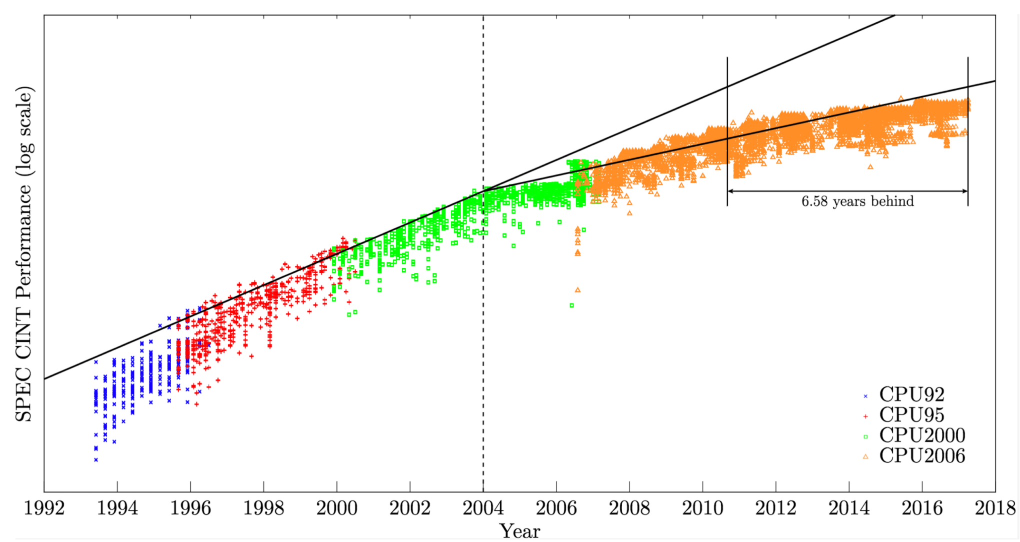
- Thursday May 3 in class
- Closed book, closed notes

Moore's Law



Source: Intel/Wikipedia

Single-Threaded Performance Not Improving

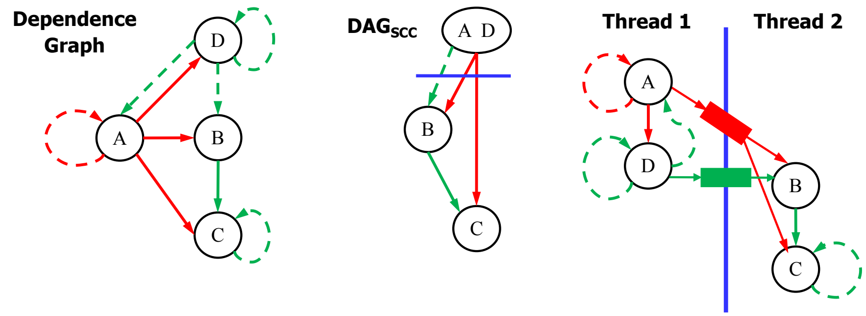


Decoupled Software Pipelining

[MICRO 2005]

Decoupled Software Pipelining (DSWP)

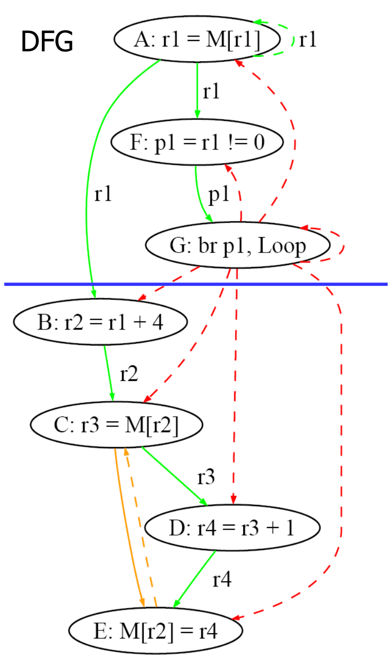
```
A: while(node)
B:   ncost = doit(node);
C:   cost += ncost;
D:   node = node->next;
```



register
control
intra-iteration
loop-carried
communication queue

Inter-thread communication latency is a one-time cost

Implementing DSWP



L1:

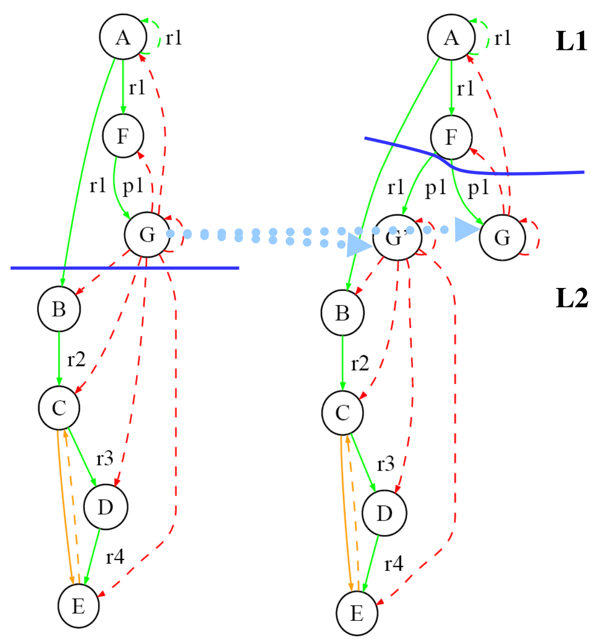
```
     SPAWN(Aux)
A:   r1 = M[r1]
     PRODUCE [1] = r1
F:   p1 = r1 != 0
G:   br p1, L1
```

Aux:

```
     CONSUME r1 = [1]
B:   r2 = r1 + 4
C:   r3 = M[r2]
D:   r4 = r3 + 1
E:   M[r2] = r4
```

register
control
memory
intra-iteration
loop-carried

Optimization: Node Splitting To Eliminate Cross Thread Control



L1

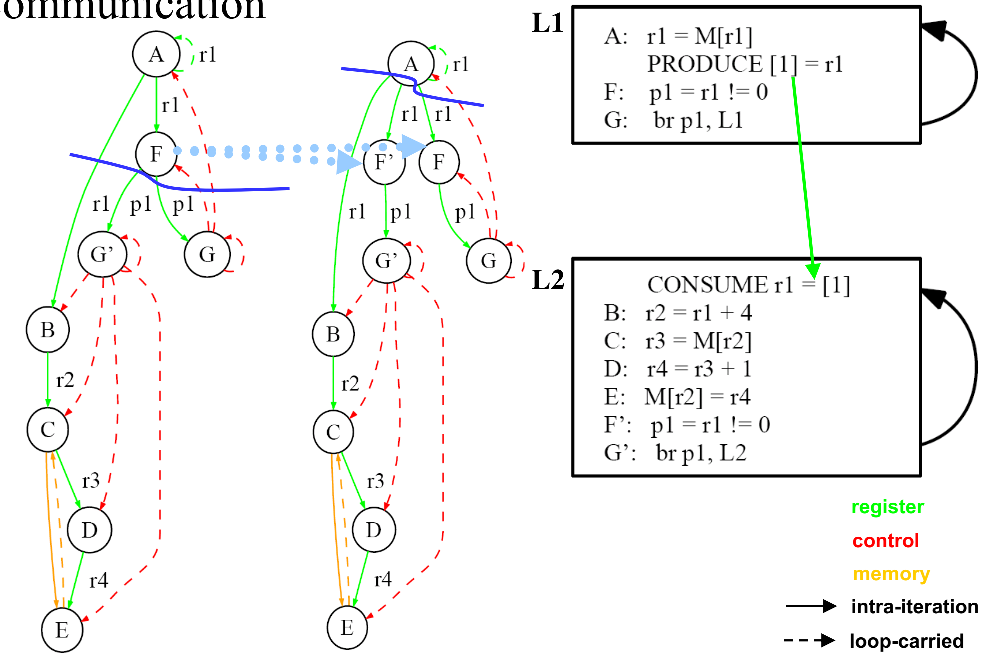
```
A:  r1 = M[r1]
    PRODUCE [1] = r1
F:  p1 = r1 != 0
    PRODUCE [2] = p1
G:  br p1, L1
```

L2

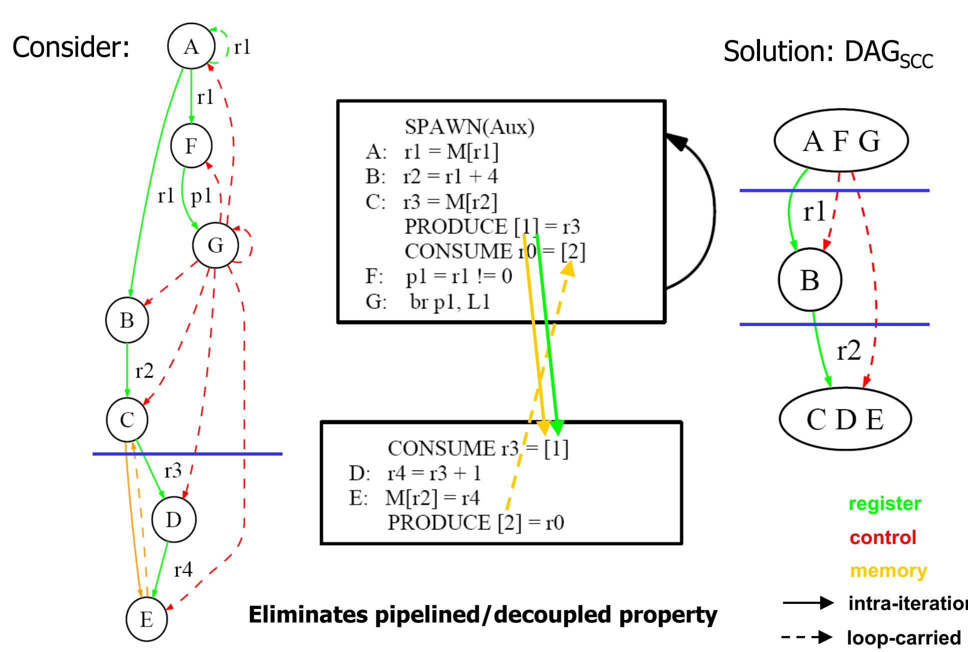
```
    CONSUME r1 = [1]
B:  r2 = r1 + 4
C:  r3 = M[r2]
D:  r4 = r3 + 1
E:  M[r2] = r4
    CONSUME p1 = [2]
G': br p1, L2
```

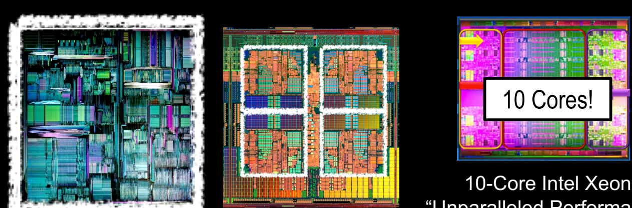
register
control
memory
intra-iteration
loop-carried

Optimization: Node Splitting To Reduce Communication



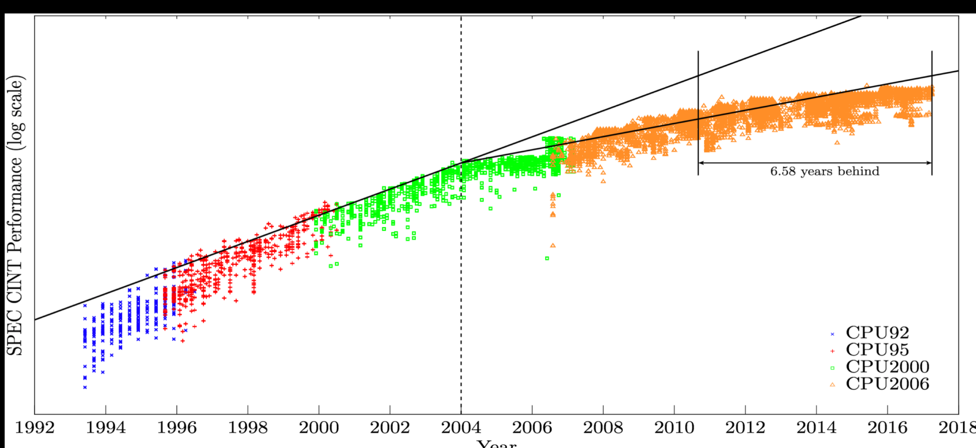
Constraint: Strongly Connected Components



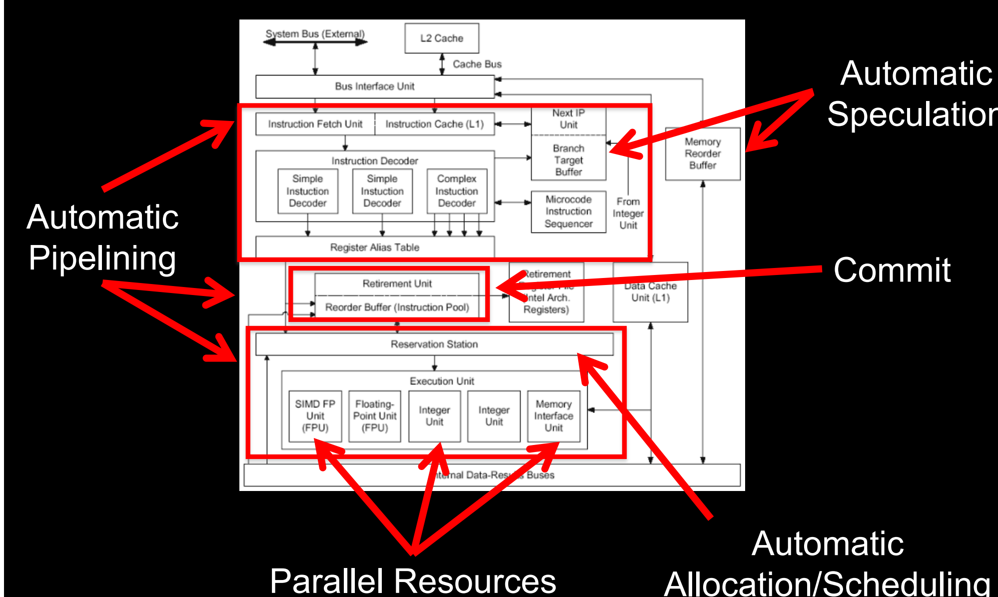


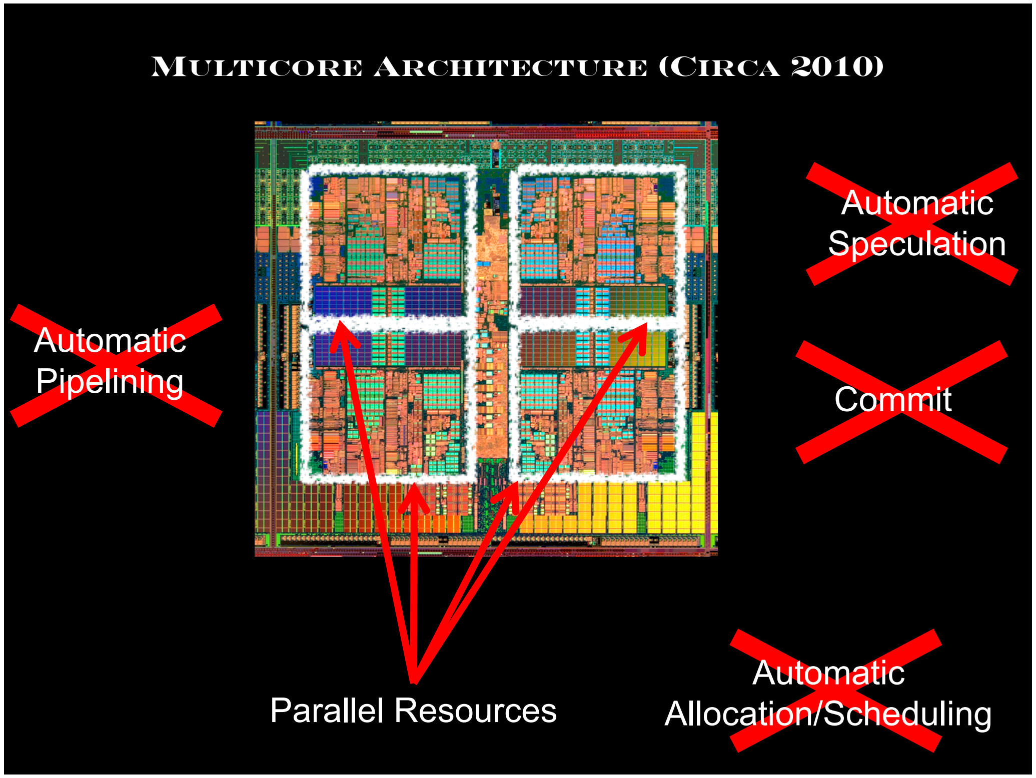
10-Core Intel Xeon
“Unparalleled Performance”

Era of DIY:
- Multicore
- Reconfigurable
- GPUs
- Clusters

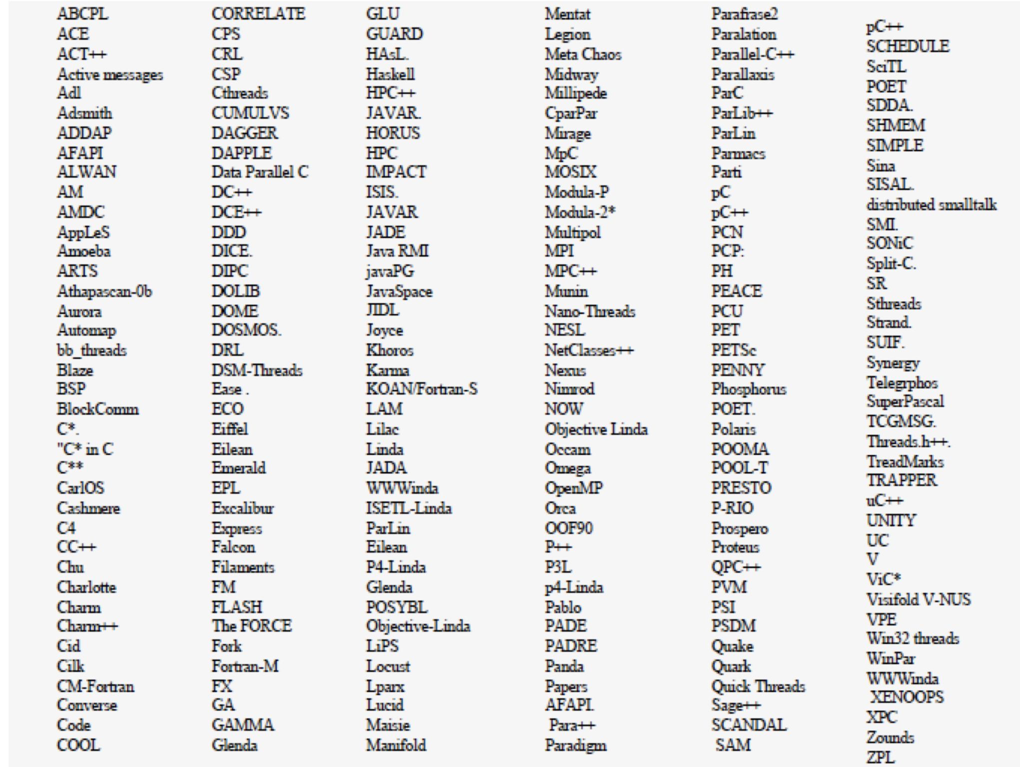


P6 SUPERSCALAR ARCHITECTURE (CIRCA 1994)

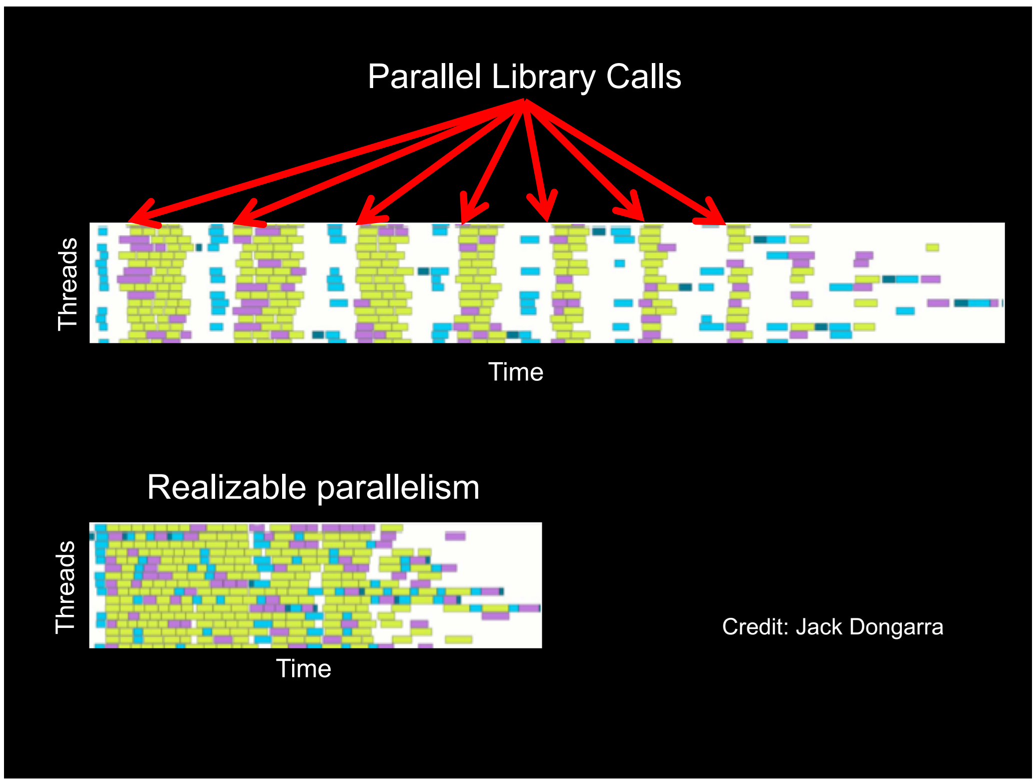




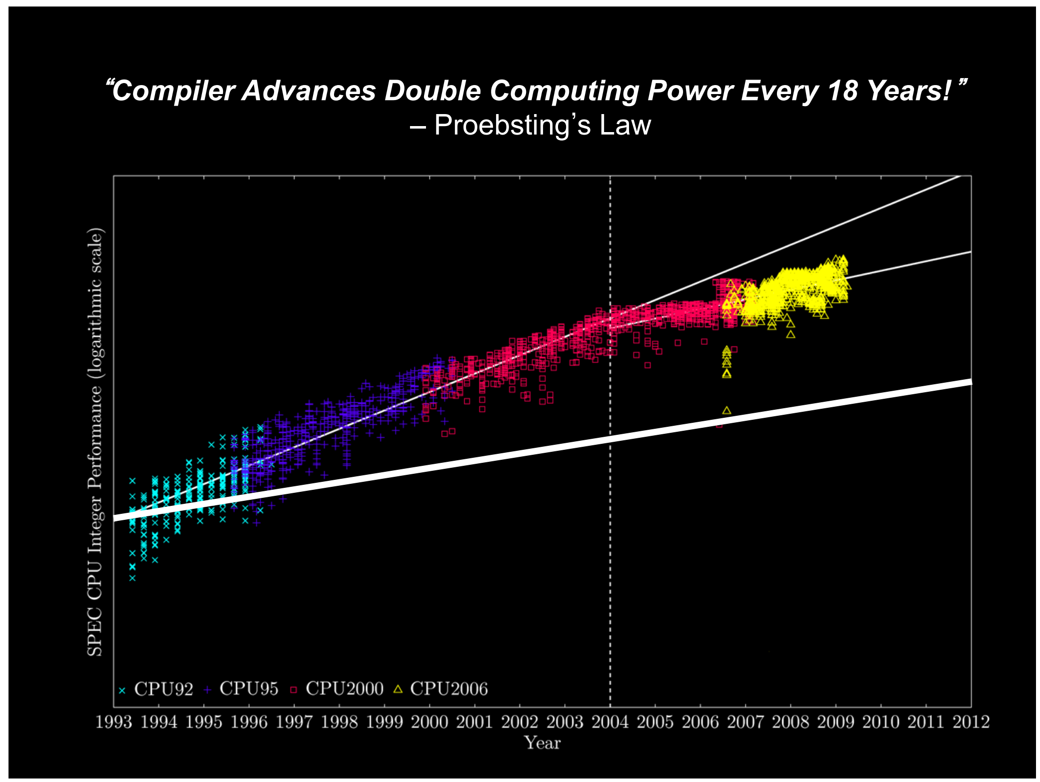
MULTICORE ARCHITECTURE (CIRCA 2010)
Automatic Speculation
Automatic Pipelining
Commit
Parallel Resources
Automatic Allocation/Scheduling



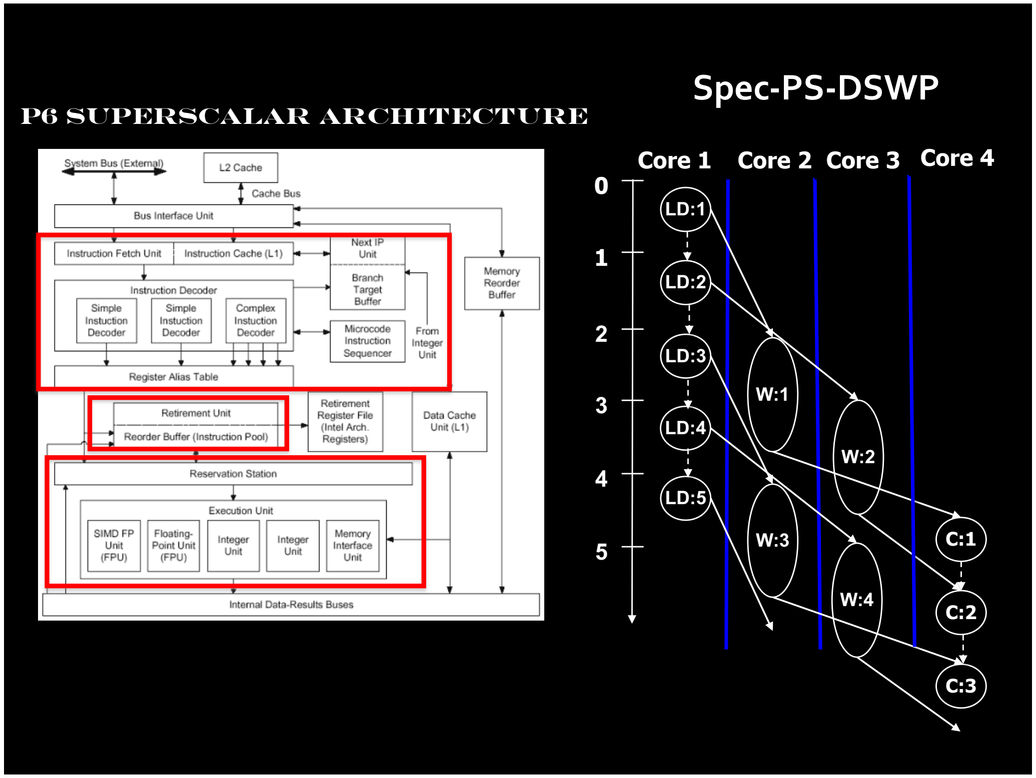
ABCPL
ACE
ACT++
Active messages
Adl
Adsmith
ADDAP
AFAPI
ALWAN
AM
AMDC
AppLeS
Amoeba
ARTS
Athapascan-0b
Aurora
Automap
bb_threads
Blaze
BSP
BlockComm
C*.
"C* in C
C**
CarlOS
Cashmere
C4
CC++
Chu
Charlotte
Charm
Charm++
Cid
Cilk
CM-Fortran
Converse
Code
COOL
CORRELATE
CPS
CRL
CSP
Cthreads
CUMULVS
DAGGER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

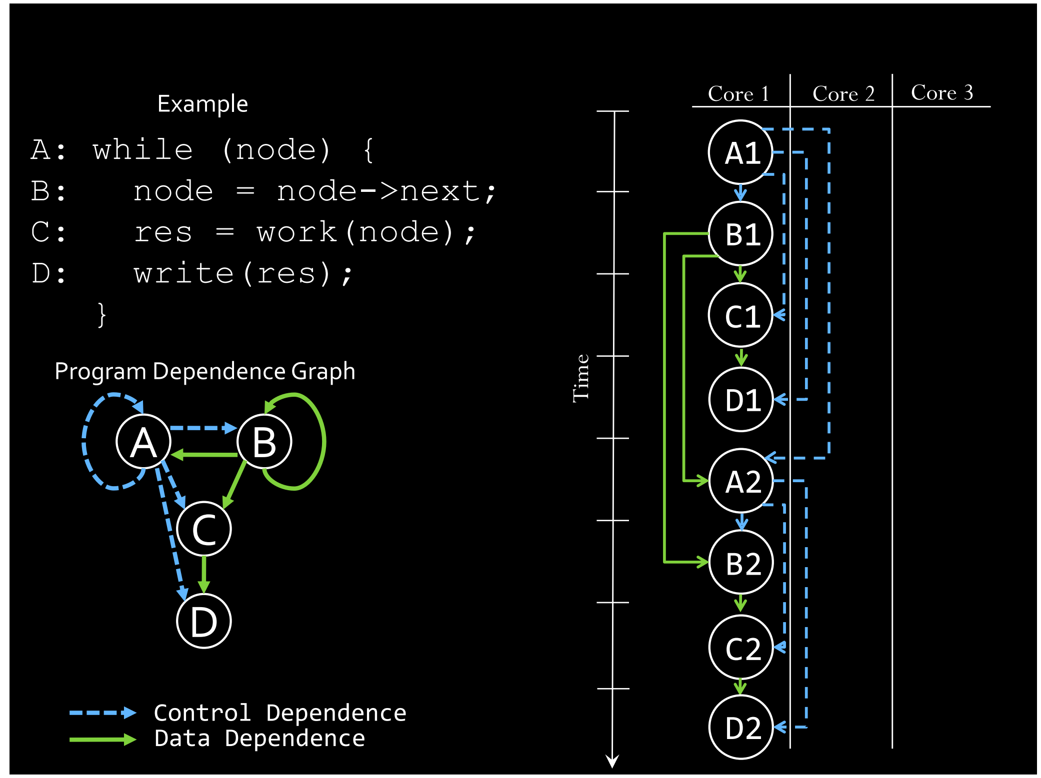


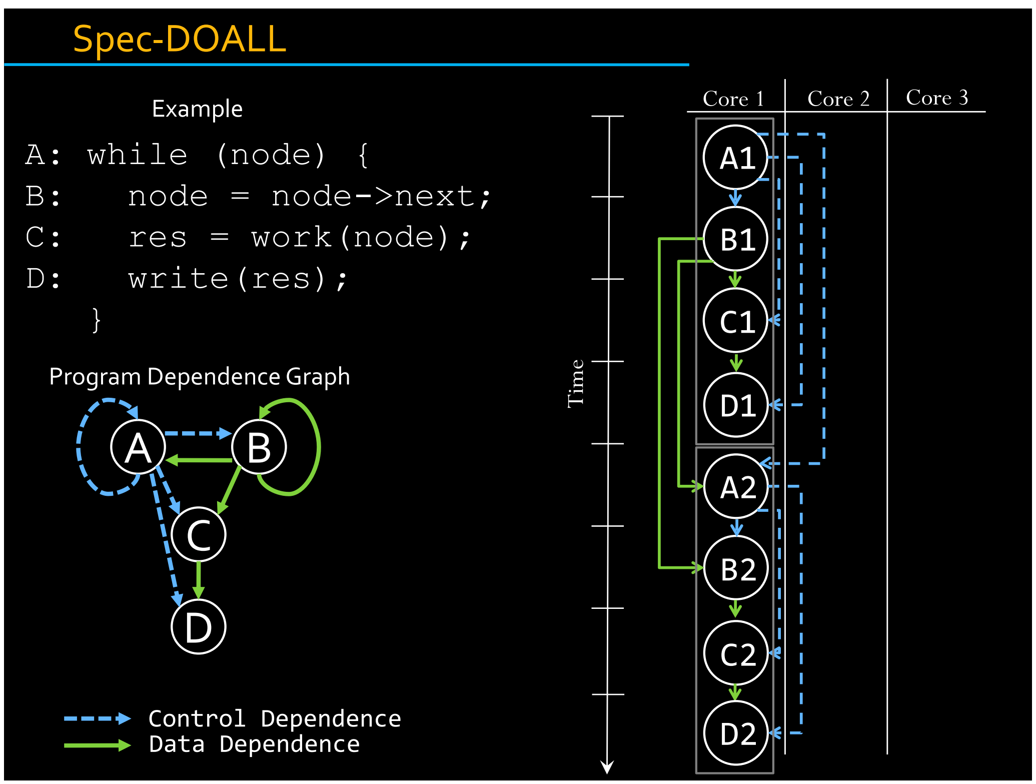
Parallel Library Calls
Threads
Time
Realizable parallelism
Threads
Time
Credit: Jack Dongarra

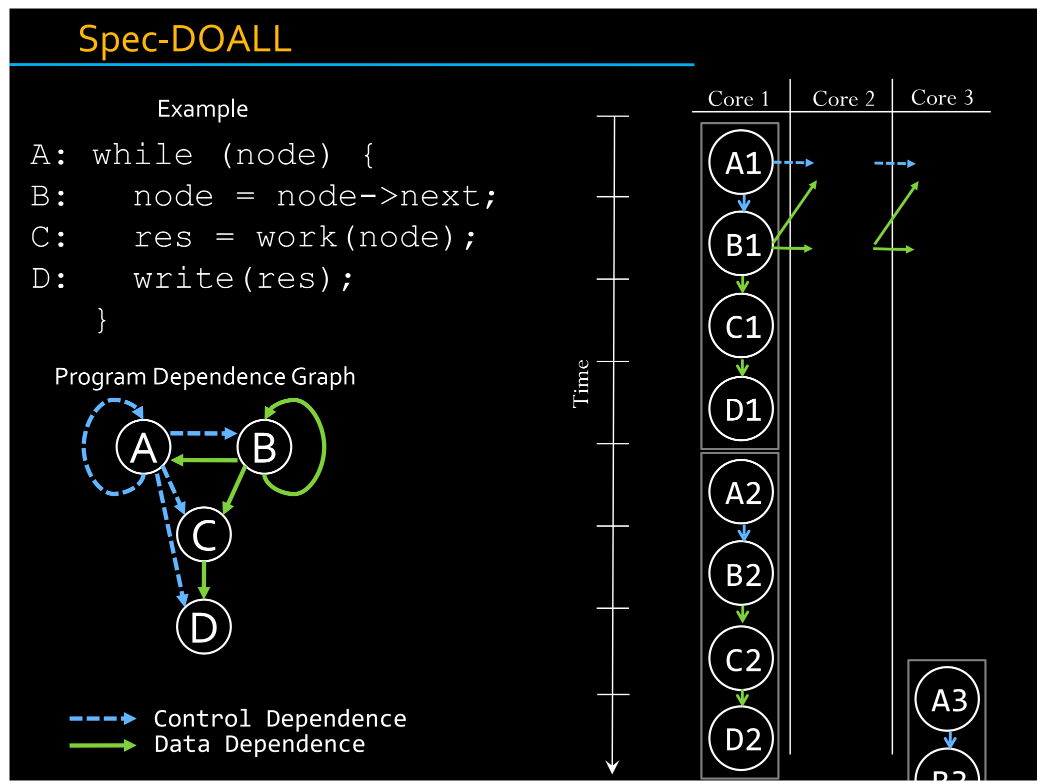


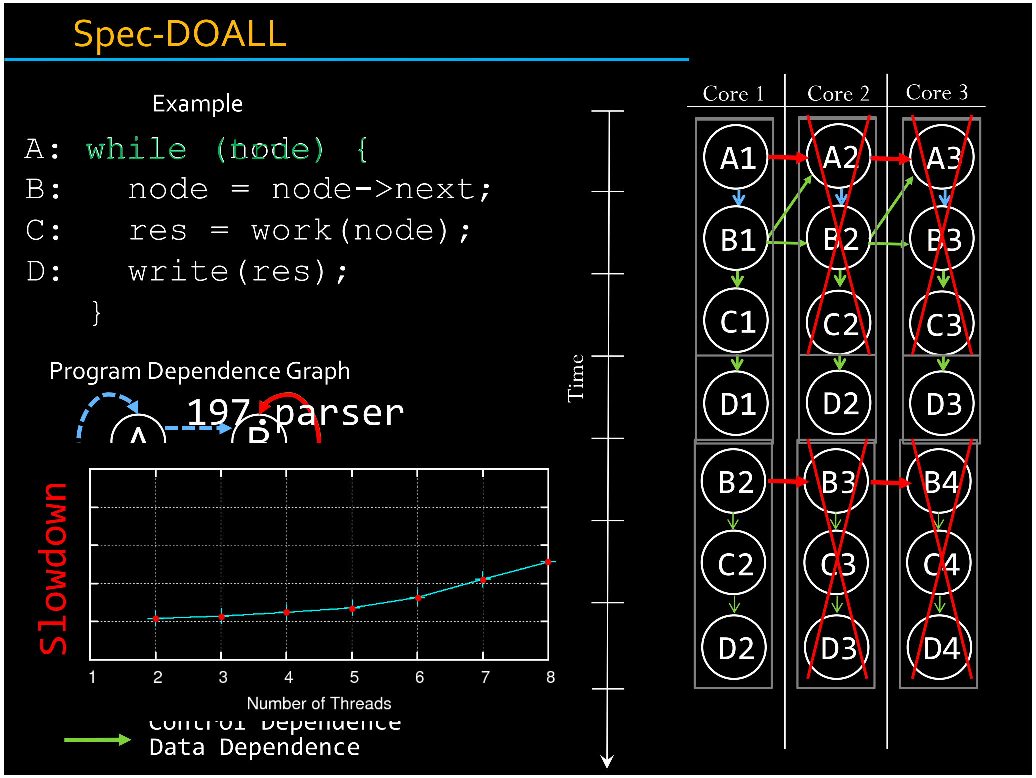
"Compiler Advances Double Computing Power Every 18 Years!"
– Proebsting's Law
SPEC CPU Integer Performance (logarithmic scale)
CPU92
CPU95
CPU2000
CPU2006
1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
Year

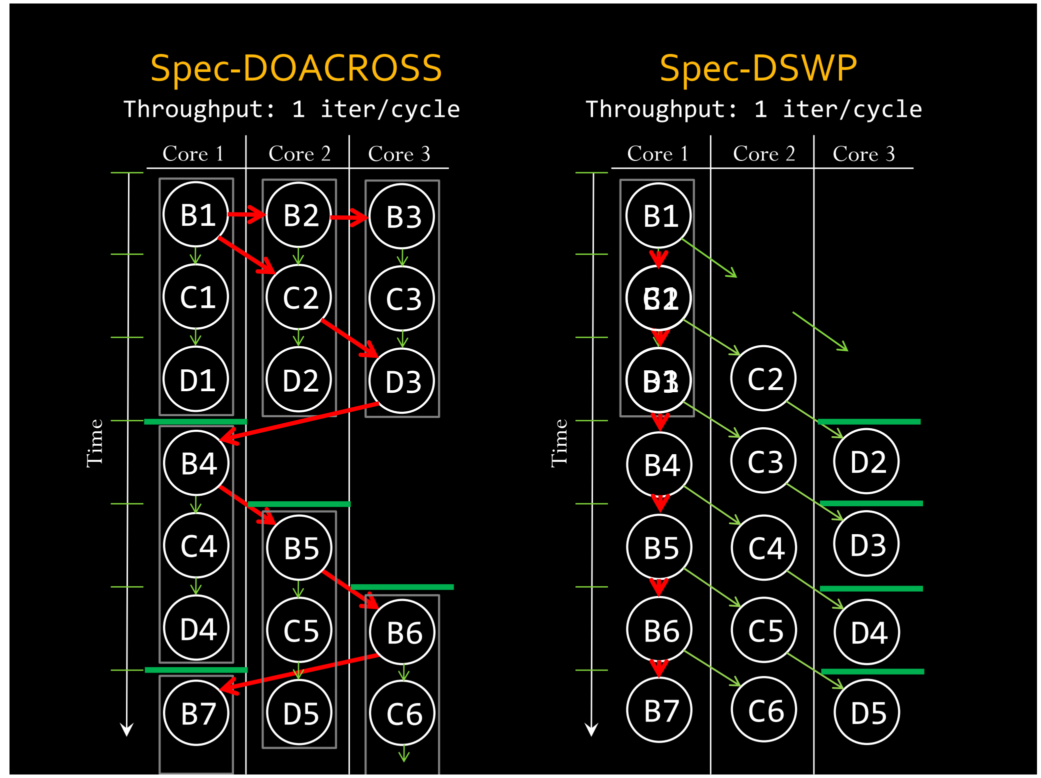

P6 SUPERSCALAR ARCHITECTURE
System Bus (External)
L2 Cache
Cache Bus
Bus Interface Unit
Instruction Fetch Unit
Instruction Cache (L1)
Next IP Unit
Branch Target Buffer
Instruction Decoder
Simple Instuction Decoder
Simple Instuction Decoder
Complex Instuction Decoder
Microcode Instruction Sequencer
From Integer Unit
Memory Reorder Buffer
Register Alias Table
Retirement Unit
Reorder Buffer (Instruction Pool)
Retirement Register File (Intel Arch. Registers)
Data Cache Unit (L1)
Reservation Station
Execution Unit
SIMD FP Unit (FPU)
Floating-Point Unit (FPU)
Integer Unit
Integer Unit
Memory Interface Unit
Internal Data-Results Buses
Spec-PS-DSWP
Core 1
Core 2
Core 3
Core 4
0
1
2
3
4
5
LD:1
LD:2
LD:3
LD:4
LD:5
W:1
W:2
W:3
W:4
C:1
C:2
C:3

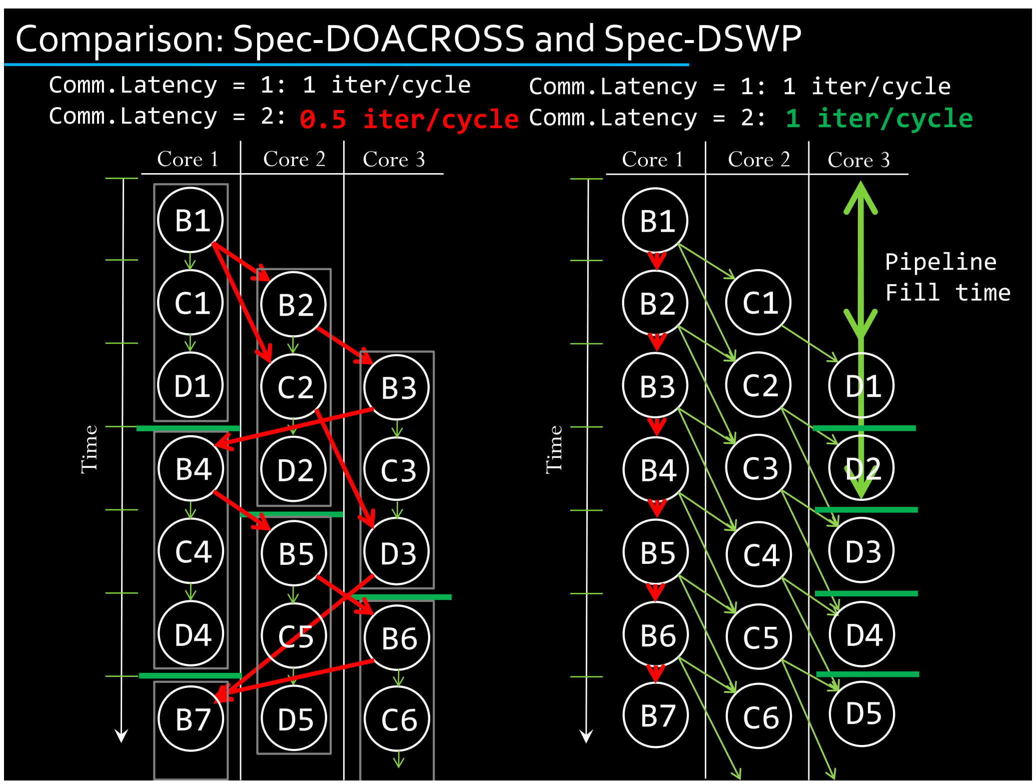

Example
A: while (node) {
B: node = node->next;
C: res = work(node);
D: write(res);
}
Program Dependence Graph
A
B
C
D
Control Dependence
Data Dependence
Time
Core 1
Core 2
Core 3
A1
B1
C1
D1
A2
B2
C2
D2

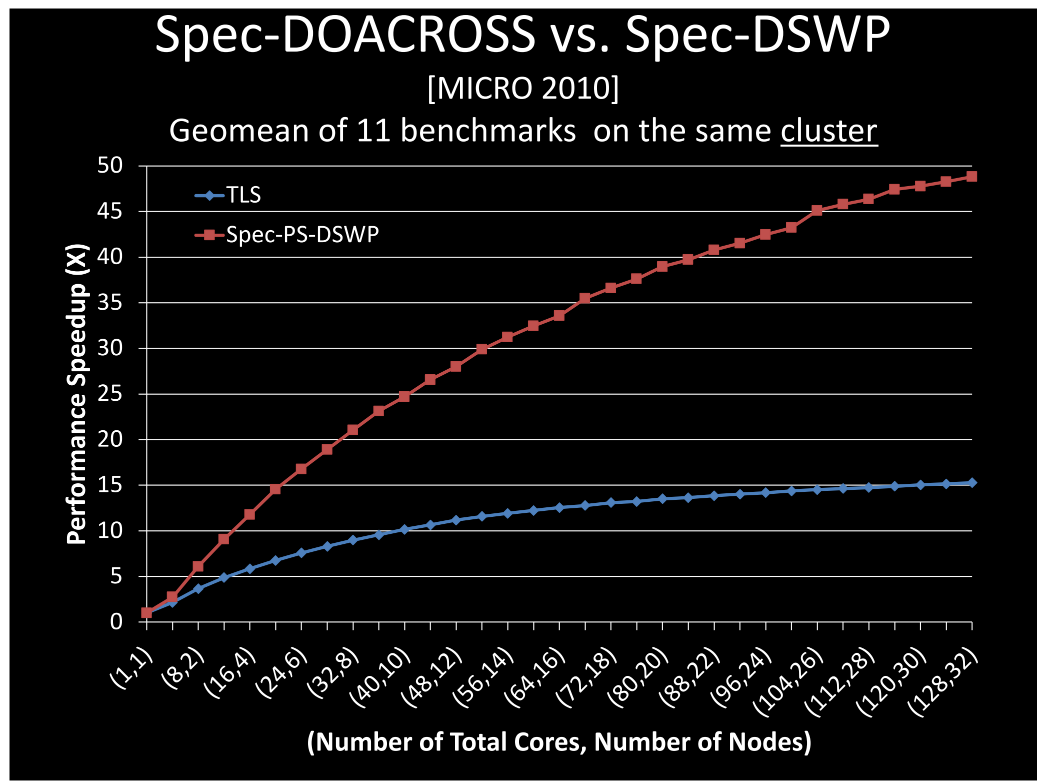

Spec-DOALL
Example
A: while (node) {
B: node = node->next;
C: res = work(node);
D: write(res);
}
Program Dependence Graph
A
B
C
D
Control Dependence
Data Dependence
Time
Core 1
Core 2
Core 3
A1
B1
C1
D1
A2
B2
C2
D2


Spec-DOALL
Example
A: while (node) {
B: node = node->next;
C: res = work(node);
D: write(res);
}
Program Dependence Graph
A
B
C
D
Control Dependence
Data Dependence
Time
Core 1
Core 2
Core 3
A1
B1
C1
D1
A2
B2
C2
D2
A3


Spec-DOALL
Example
A: while (node) {
B: node = node->next;
C: res = work(node);
D: write(res);
}
Program Dependence Graph
197.parser
Slowdown
1 2 3 4 5 6 7 8
Number of Threads
Data Dependence
Time
Core 1 Core 2 Core 3
A1 A2 A3
B1 B2 B3
C1 C2 C3
D1 D2 D3
B2 B3 B4
C2 C3 C4
D2 D3 D4


Spec-DOACROSS
Throughput: 1 iter/cycle
Spec-DSWP
Throughput: 1 iter/cycle
Core 1 Core 2 Core 3
Time


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
Core 1 Core 2 Core 3
Time
Pipeline
Fill time


Spec-DOACROSS vs. Spec-DSWP
[MICRO 2010]
Geomean of 11 benchmarks on the same cluster
TLS
Spec-PS-DSWP
Performance Speedup (X)
0 5 10 15 20 25 30 35 40 45 50
(1,1) (8,2) (16,4) (24,6) (32,8) (40,10) (48,12) (56,14) (64,16) (72,18) (80,20) (88,22) (96,24) (104,26) (112,28) (120,30) (128,32)
(Number of Total Cores, Number of Nodes)