.

Instruction Level Distributed Processing:

December 2000

J. E. Smith

Dept. of Elect. and Comp. Engr.
1415 Johnson Drive
Univ. of Wisconsin
Madison, WI 53706

jes@ece.wisc.edu
http://www.engr.wisc.edu/ece/faculty/smith_james.html

Evolution of High-End Processors

- Focus has been on Performance

  serial

  -> pipelining

  -> superscalar

  -> wide superscalar

- Larger, Wider, More!

  wider fetch

  wider issue

  larger windows

  more outstanding memory ops

  more speculation

  => *Increasingly heavyweight microprocessors*

Past (and Current) Architecture Trends

- Emphasis has been on *exploiting* technology

  (especially increasing transistor counts)

  => many transistors

  => increasing intellectual complexity

- Architecture must also *accommodate* technology shifts

  e.g. cache memories
  -- adjust to shifts in proc./RAM technology

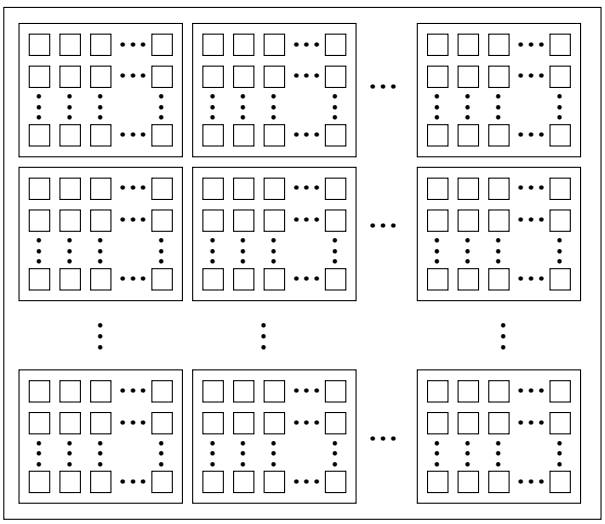
- Major shifts occurring now:

  Wire vs. transistor delays

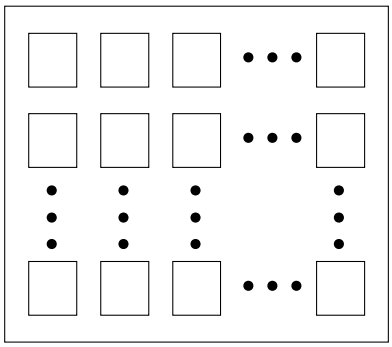
  Power vs. performance

  Application shifts

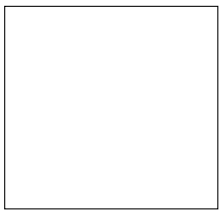
Computer Evolution: from a packaging prespective



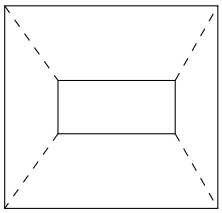
Mainframe



Minicomputer



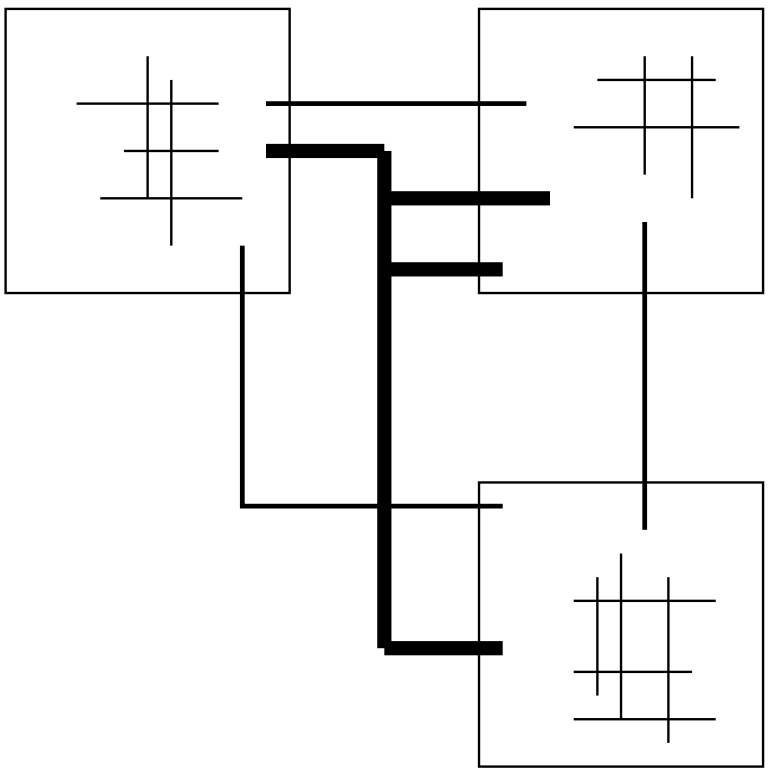
Microprocessor



Core processor

Future Processors: Wires

- Local wires: high congestion
- Global wires: long delay
- Global multi-drop buses: very long delay



Wire Delays

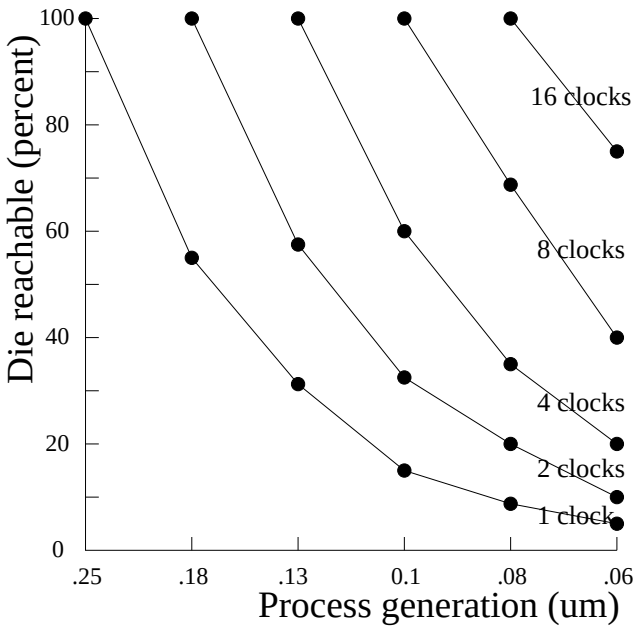
- Wire delays will not scale well

  Wire aspect ratios will reach a limit

  => Resistance goes up even as length

  Fringing capacitance will dominate diminishes

- Limited chip reachability (Matzke 9/97 Computer)



- Limited on-chip RAM reachability

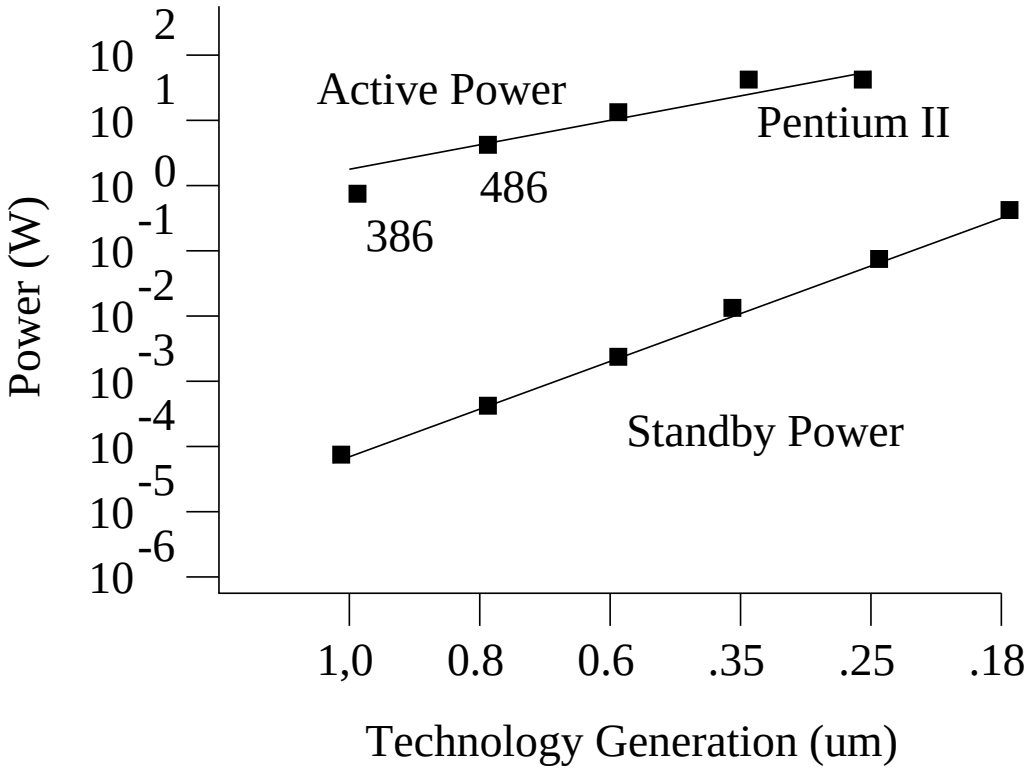
  For 35 nm technology, in a single cycle, half the RAM will be reachable as today

  (Agarwal et al., 2000 ISCA)

Power Issues

- Dynamic power has become a very big issue even for high-end processors

- But it gets worse...



(from Thompson Packan and Bohr -- Intel -- 1998.)

- Static power solutions:

  selective power-gating

  use fewer transistors

  use mostly slower transistors (less leakage)

Applications and Software

- Many high-end server applications are becoming throughput oriented -- e.g. web servers.

  => shift in general purpose computing
  toward multiple latency-limited threads

- Many (most?) programs are not highly optmized

  Dynamic linking reduces opportunities

  Optimization takes time/effort

  Optimization intoduces/exposes bugs

  Multiple platforms

- Many highly parallel tasks no longer performed in general purpose processors

  Library-based VLIW/vector implementations

  faster for many high ILP applications

  also more energy efficient

Future Processors: WWSCD?

What Would Seymour Cray Do?

- Keep it simple
- Fast clocks
  - -- one sure way of improving overall perf.
  - -- a key to S. Cray's success
  - -- evident in Pentium evolution

P5 5 stages

Pref	Dec 1	Dec 2	Exec	Wrt Bck

P6 12 stages

F1	F2	D1	D2	D3	Rn	ROB	Rdy/Sch	Disp	Ex	Ret1	Ret2

Willamette 20 stages

IP1	IP2	TC1	TC2	Dr	Al	Rn	Q	S1	S2	S3	Dp1	Dp2	R1	R2	Ex	Flgs	Br	Dr

WWSCD: important note

SC used relatively short pipelines
and controlled the instruction set
and didn't predict/speculate
and didn't obsess over issue slots

Cray-1 5 stages

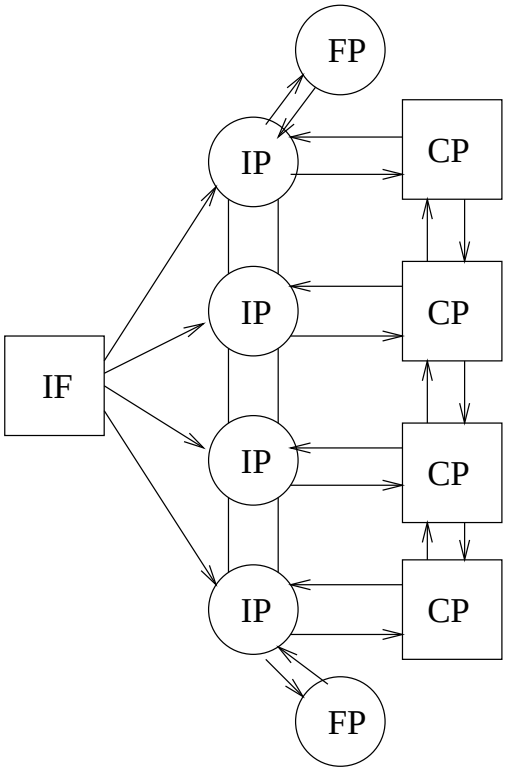
If	Dec	Is	Ex1	Ex2	Wb

Cyber 180/990 15 stages

Adr	If	Prc	Dr	Map	Dr	uC	D1	nC	Buf	Reg	Ex1	Ex2	Wb

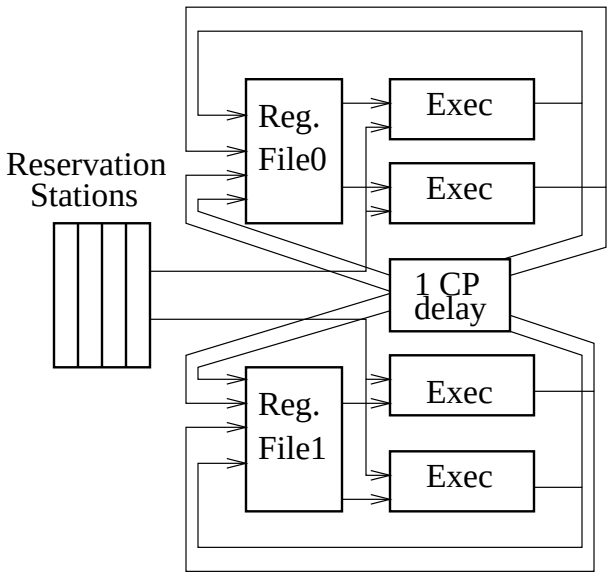
Implications: Microarchitecture

- Instruction Level Distributed Processing (ILDP)
  - --blocks with few transistors running very fast
  - --point-to-point global interconnects
  - --little speculation (only br. predict and caches)
  - --determinism is better than arbitration



=> microprocessors built of *distributed*, *fast*, *light-weight* elements

ILDP Precursor: Alpha 21264



- Leading edge designs hit limits first
- Inter-cluster bypass => 1 cycle
- Dependences will draw communicating instructions to same cluster

Dependence-based Microarchitecture

- Goals:

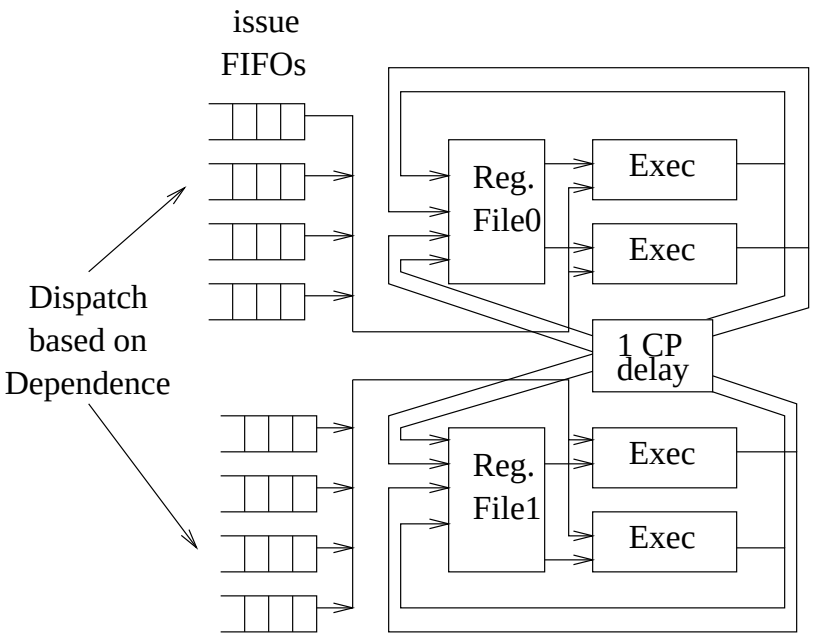
  Simplify issue logic

  Accommodate wire delays

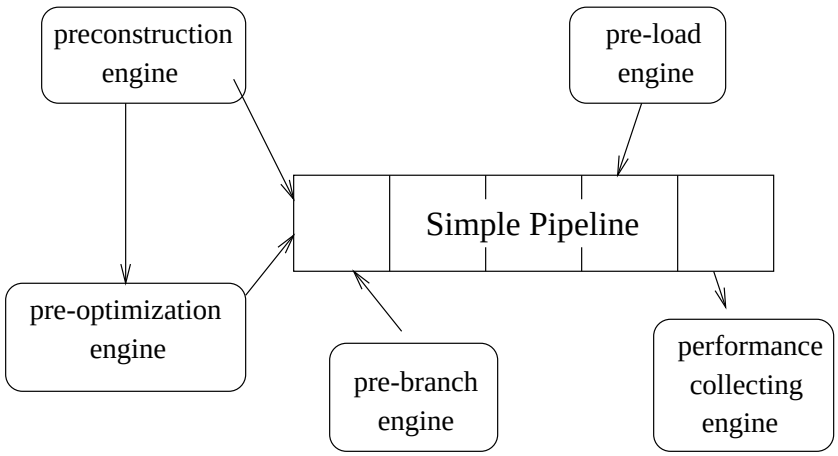
- Cluster organization

- Steer instructions at dispatch time

- Issue from multiple FIFOs

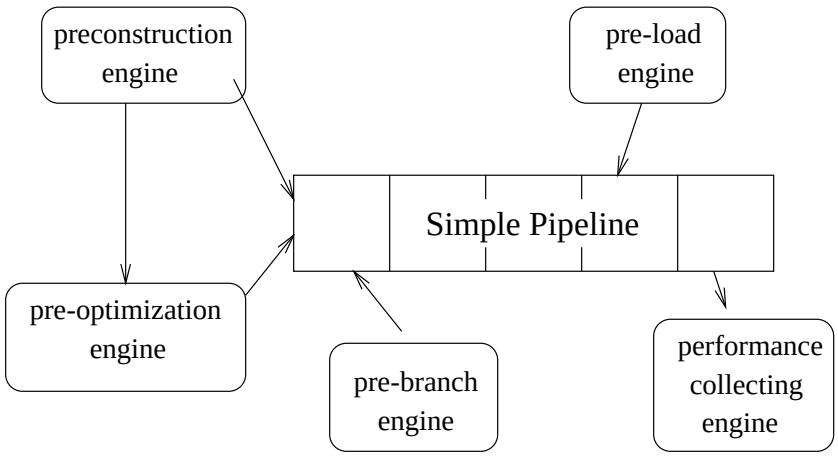


Heterogeneous ILDP



- Simple processing pipelines for high performance
- Separate, outlying "helper engines"
- Non-critical communication delay between pipeline and helper engines
- Helper engines can use slower (less leaky) transistors

Helper Engines



- Pre-load engines (Roth & Sohi, ISCA 99)
- Branch engines (Reinman, et al. ISCA 99)
- Performance profiling engines (Heil Micro 00)
- Instruction Co-Processor (Shen, ISCA 00)
- Checking Engines (Austin Micro 99)

Implications: Chip Architecture

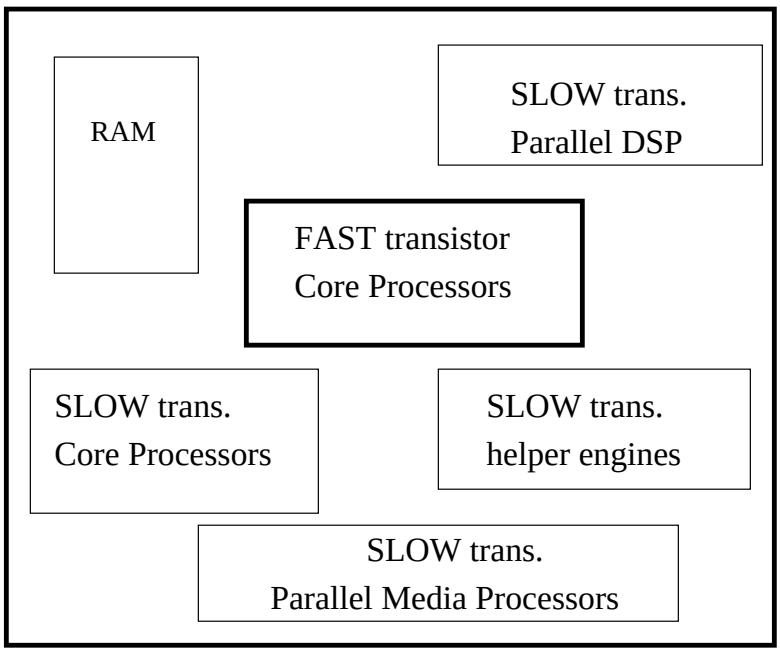
- *Cores may become a solved problem*

  few transistors/emphasis on fast clock
  => fewer opportunities for complications

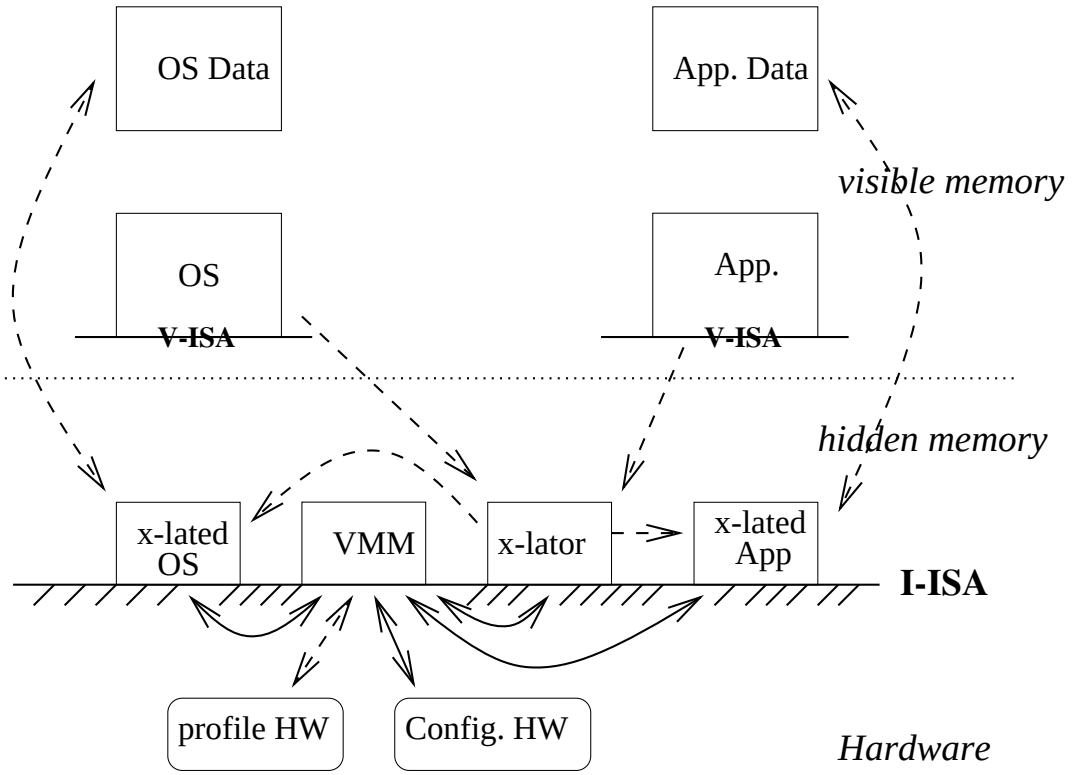
- Emphasis may shift to chip architecture

- Remove high ILP media/DSP functions from core (into slow, parallel, non-leaky transistors?)

- Perf. optimization with slow helper processors



Implications: co-Designed Virtual Machines



- Gives hardware designers a layer of software
- Can include binary translation
- Allows dynamic optimization/adaptation
- Orchestrates ILDP
- Manages on-chip resources

  (IBM Daisy & BOA; Transmeta)

Co-Designed VMs: Orchestrating ILDP

- Simplified control of ILDP

  Some elements of program structure must be known by hardware

    e.g. data flow graph

    control flow graph

    register value locality

  Hardware has a *restricted* view of the program

- ISA flexibility

  Rich variety of ILDP microarchitectures

  Developing ISA for a specific microarchitecture is often a mistake

- Dynamic profiling/feedback also beneficial

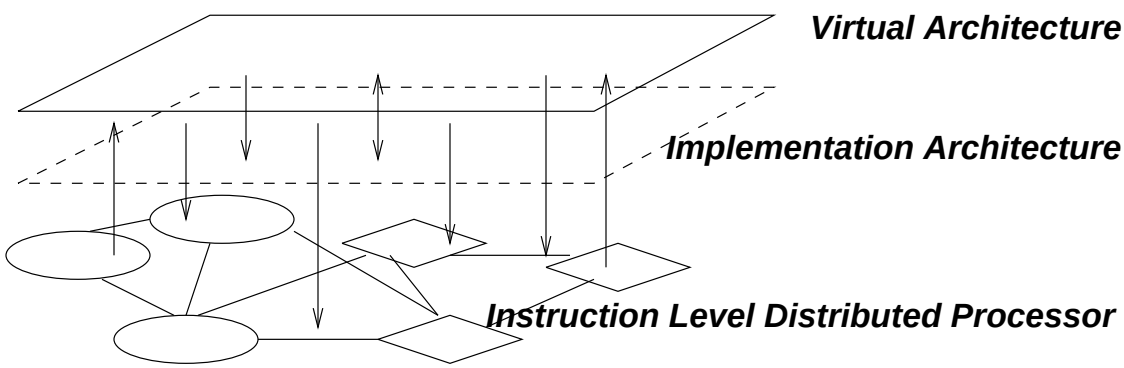
  Some communication patterns may only be apparent at runtime

- Other applications

  High level power management

  High level fault tolerance

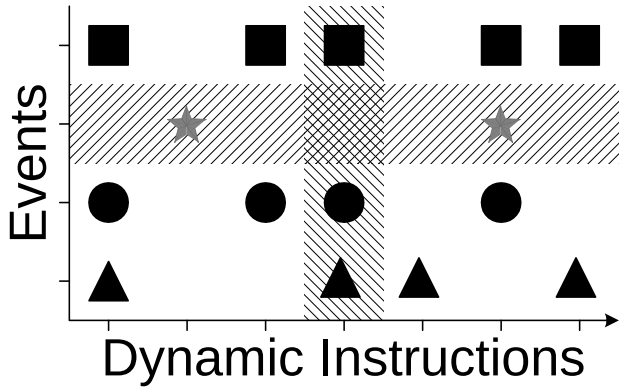
co-Designed Virtual Machines: the Big Picture



A Profiling Engine

- Profiling may become an integral part of ILDP
- For dynamic software optimization
- For dynamic hardware adaptivity
- Should include generality and flexibility
- Should support current (and future) applications

Relational Profiling Model



- Instruction based queries

  "For certain instructions what events occurred?"

- Event based queries

  "For some events what instructions were involved?"

- Hybrids

  "For all load instructions that hit in the cache..."

Example Queries

- Applications:

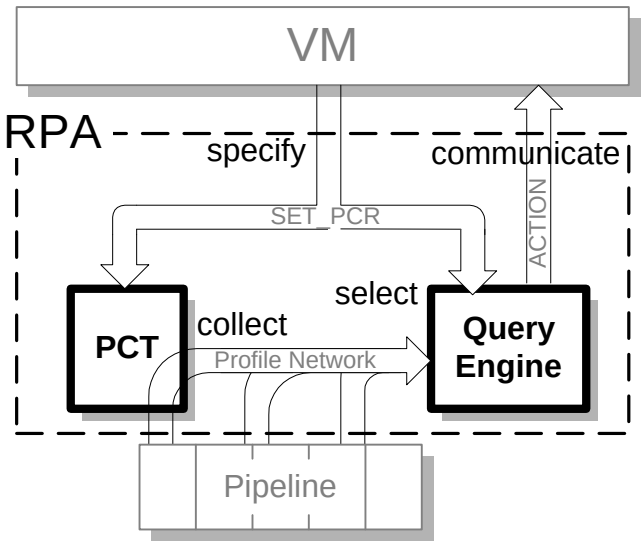
  Edge profiling -- profile conditional branches

  Prefetching -- profile cache misses

  Concurrent garbage collection

    store barriers -- detect stores of references to region being collected

Implementation



- PCT -- profile control table

  sampling rate, info to be collected

- Network -- paths on which info flows

- Query engine

  performs matches for requested info

  initiates actions -- traps, counters, messages

- More info in Micro 00

Implications: Instruction Sets

- Optimized for ILDP
- Focus on *communication* and *dependence*
  NOT computation and *independence*
  (in contrast to VLIW)
- Optmize for fast *execution*
  NOT decode
  (in contrast to RISC)
  => small, fast structures
  caches
  register files
- Small instruction footprint
  Variable length instructions ?
- Enabled by transparent binary translation
  e.g. IBM System/38, AS/400; Daisy; Transmeta

Example ISA

- Consider an accumulator-based ISA
- Perhaps 64 GPRs
- Implementation in clusters each w/ Accumulator
- Intra-cluster communication via Accumulator
- Inter-cluster communication via GPRs

```
R <- A                1 byte
A <- R                1 byte
A <- A op R           2 bytes
A <- A op imm         4 bytes
A <- M(R op imm)      4 bytes
M(R op imm) <- A      4 bytes
R <- M(A op imm)      4 bytes
```

etc.

<< ideas borrowed from Cray-1S IOP >>

Summary and Conclusions

- Technology will force shifts in a number of areas
- Microarchitectures
  - Instruction level distributed processing
  - Simple/fast core processors
  - Distributed resources => simpler management
- Chip architecture
  - Will become focus of architecture research
  - Distribution of functions will be key
- Virtual Machines
  - Provide hardware designers w/ layer of software
  - Enables a rich variety of ISAs and implementations
- Instruction Sets
  - Its time to re-think instruction sets (again)
  - Focus on communication and dependence

This work is supported the National Science Foundation, IBM, SUN Microsystems, Intel.