Intel Developer Forum
Fall 2001

Multi-Threaded Programming for Next Generation Multi-Processing Technology

Shihjong Kuo

Strategic Planning/SW Strategy

Intel Corporation

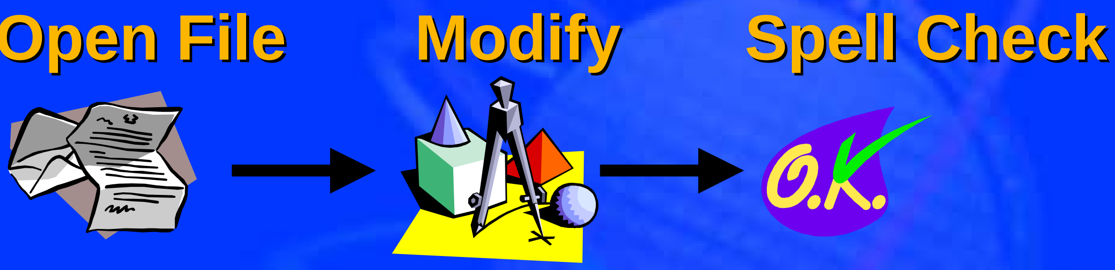
August, 2001

Agenda

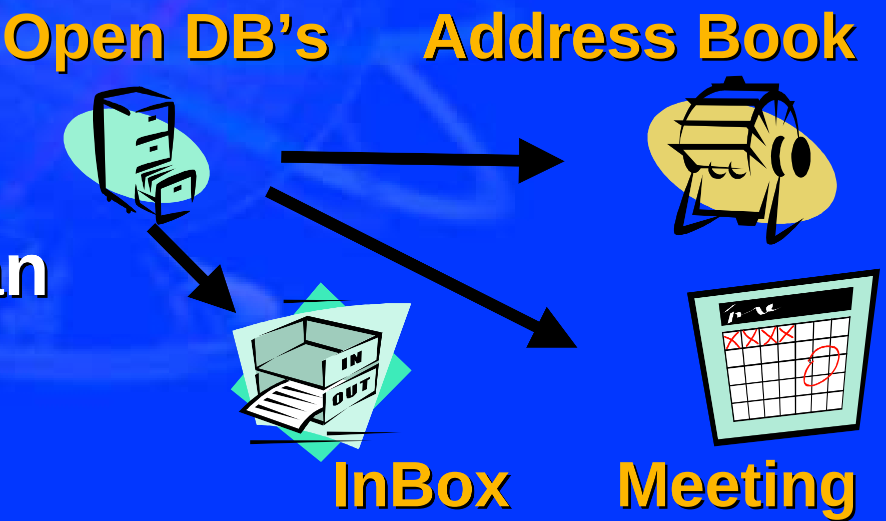
- Multi-threading (MT) and Hyper-Threading Technology
- Managing Threads and Resources
- Techniques for Programming MT Applications
- Examples of Hyper-Threading Technology performance

Multi-threading (MT)

- Sequential tasks



- Parallel tasks

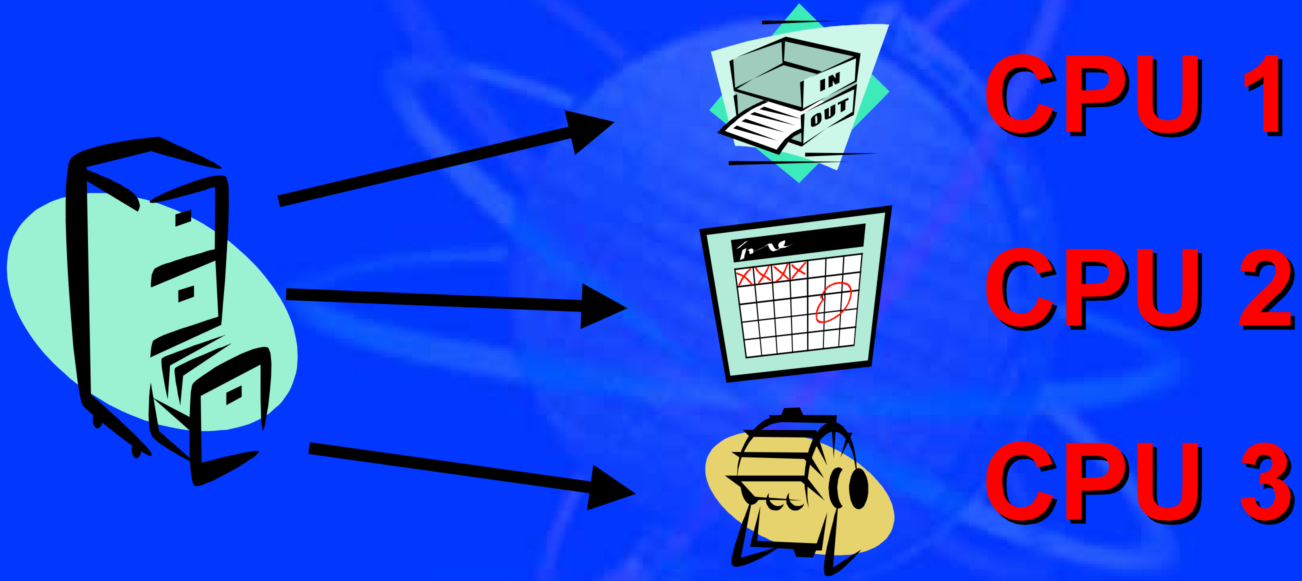


- MT applications can run on single processor system

Intel Developer Forum
Fall 2001

Multi-Processing

- Run parallel tasks using multiple processors

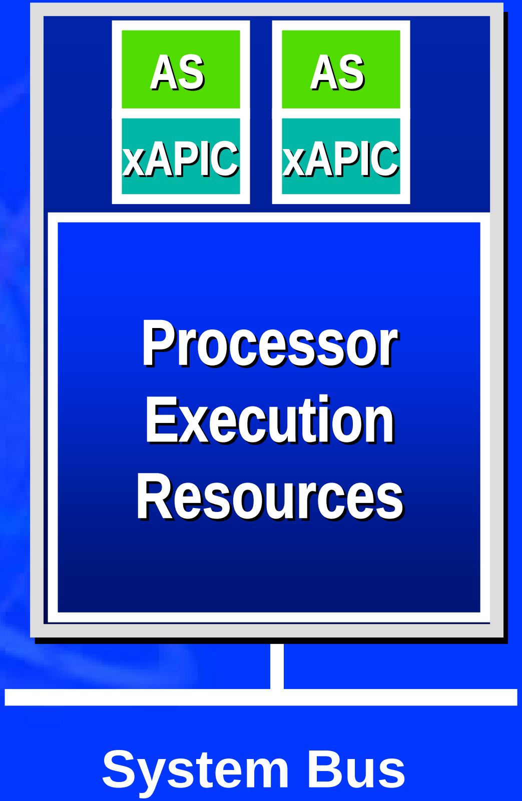


Multi-tasking workload + processor resources => Improves MT Performance

Intel Developer Forum
Fall 2001

Hyper-Threading Technology

- 2 logical processors share on-chip resources
- Increase utilization of idle resources
- 2 logical processor != 2 Physical Processors



Software sees Hyper-Threading technology as 2 processors

Intel Developer Forum Fall 2001

Design Thread Behavior for Performance and Throughput

- Data vs. Functional domain
- Compute vs. Memory bound
- Thread Synchronization

Intel Developer Forum
Fall 2001

Workload Characteristics

- Compute-bound
- Memory-bound
- Data decomposition threading
- Functional domain threading

Uncover parallelism in workload characteristics

Intel Developer Forum
Fall 2001

Managing Threads

- Synchronization:
  - Can reduce overall performance
  - Spin loops are not free
  - Idle threads should give up resources
  - Pipeline spin locks

Workload parallelism pays and make thread synchronization painless

General Guidelines to Good Performance

- Balance computation and memory operations

Basic MT Techniques Still Apply

- Use thread pools
- Don't use too many active threads
  - # of ready threads = # of processors
- Use coarse grain vs fine grain threads
- Minimize synchronization
- Don't 'False-Share' cache lines

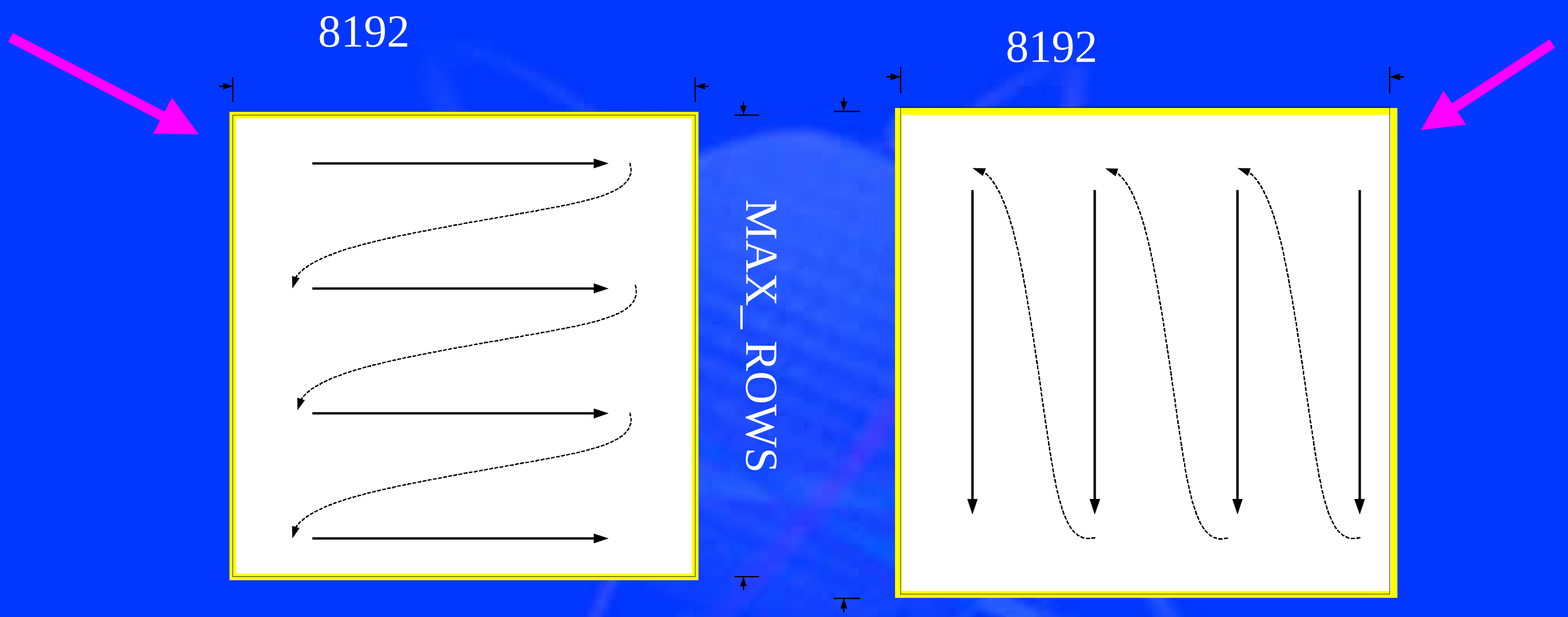
Intel Developer Forum Fall 2001

Inefficient Data Access Pattern

```
#define       MAX_ROWS        8192
#define       MAX_ COLS       8192
for ( iy = 0; iy < MAX_ROWS ; iy++) {
  for(jx = 0; jx < MAX_COLS ; jx++ )     {
     pDst [ MAX_ COLS*(jx +1) - iy – 1 ] =
                        pSrc [ iy* MAX_ROWS + jx ] ;
   } // transpose one byte at a time
}
```

Intel Developer Forum
Fall 2001

Poor Data Access Pattern



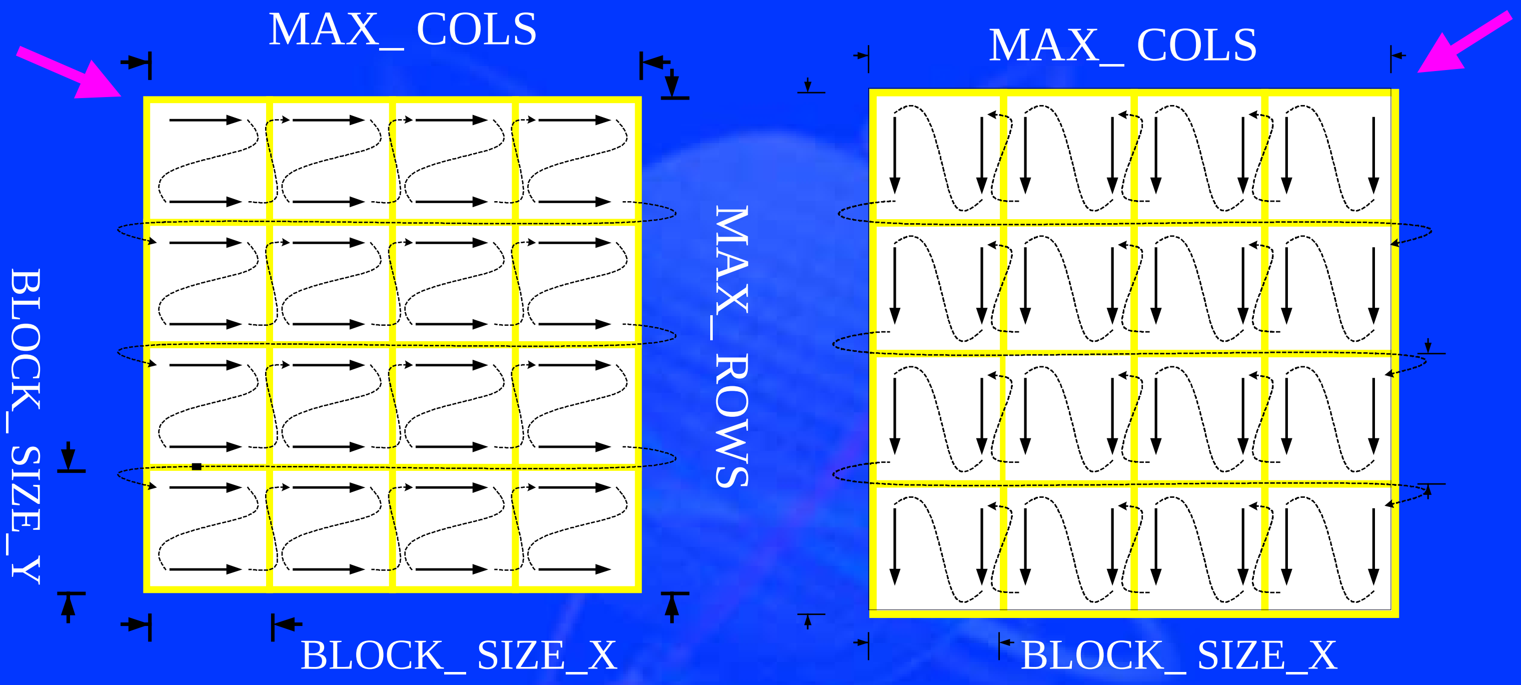
Source Array

Destination Array

Continual cache misses, excessive bus transactions

Intel Developer Forum
Fall 2001

Improving Data Access Pattern



Source Array

Destination Array

Blocking improve cache efficiency, conserve bus bandwidth

Transpose Array w/ Cache Blocking

```
#define MAX_COLS      8192
#define MAX_ROWS      8192
#define YBLOCK_SIZE  32
#define XBLOCK_SIZE  32
for (j=0; j < NUM_YBLOCKS; j++)  {
   for (k=0; k < NUM_XBLOCKS; k++) {
        yblock_beg = j*YBLOCK_SIZE;
        for (i = yblock_beg ; i < min(YBLOCK_SIZE+ yblock_beg, MAX_ROWS ); i++) {
            cols_per_block = min(XBLOCK_SIZE, MAX_COLS  - k*XBLOCK_SIZE );
            offset_src = i*MAX_COLS + k*XBLOCK_SIZE;
            offset_dst = k*MAX_COLS*XBLOCK_SIZE - i - 1;
            for ( x = 0; x < cols_per_block; x++) {
                     yinc += MAX_COLS;
                     pDst[offset_dst+yinc] = pSrc[offset_src+x];
} }  }   }
```

More code, but much faster!

Intel Developer Forum
Fall 2001

Synchronizing Short Tasks

Spin-wait loop without PAUSE

```
spin_loop:
                cmp 0, sync_var
                jne spin loop
                ..
// Spin loop w/o pause consume more
// hw resources
```

Exiting this spin-wait loop cost performance

Instructions

- HLT
  - Only executing logical processor halts
  - Increase performance of active logical processor
- PAUSE
  - Backward-compatible
  - Improves performance of spin-wait loops
- CPUID
  - Number of logical processors
  - Logical processor mapping

Intel Developer Forum
Fall 2001

Synchronizing Short Tasks

Spin-wait loop with PAUSE

```
spin_loop:
  pause        ; hint to prevent exit penalty
  cmp 0, sync_var
   jne spin_loop
```

“PAUSE” reduces delays in exiting spin loop

Intel Developer Forum
Fall 2001

Don’t Keep Idle Loop Spinning

```
// Create thread pool and suspend them
Num_created = create_threadpool( num_threads);
for (jj = 0; jj < Num_created; jj ++) {
  ResumeThread(thread_handles[jj]);
}
While (all_task_done != processor_mask ) {
  //  don’t rely on pause alone
   //  to keep idle thread spinning
  _asm pause
 …
} // spin loop consumes hw resources
```

Spin loop is not free

Intel Developer Forum Fall 2001

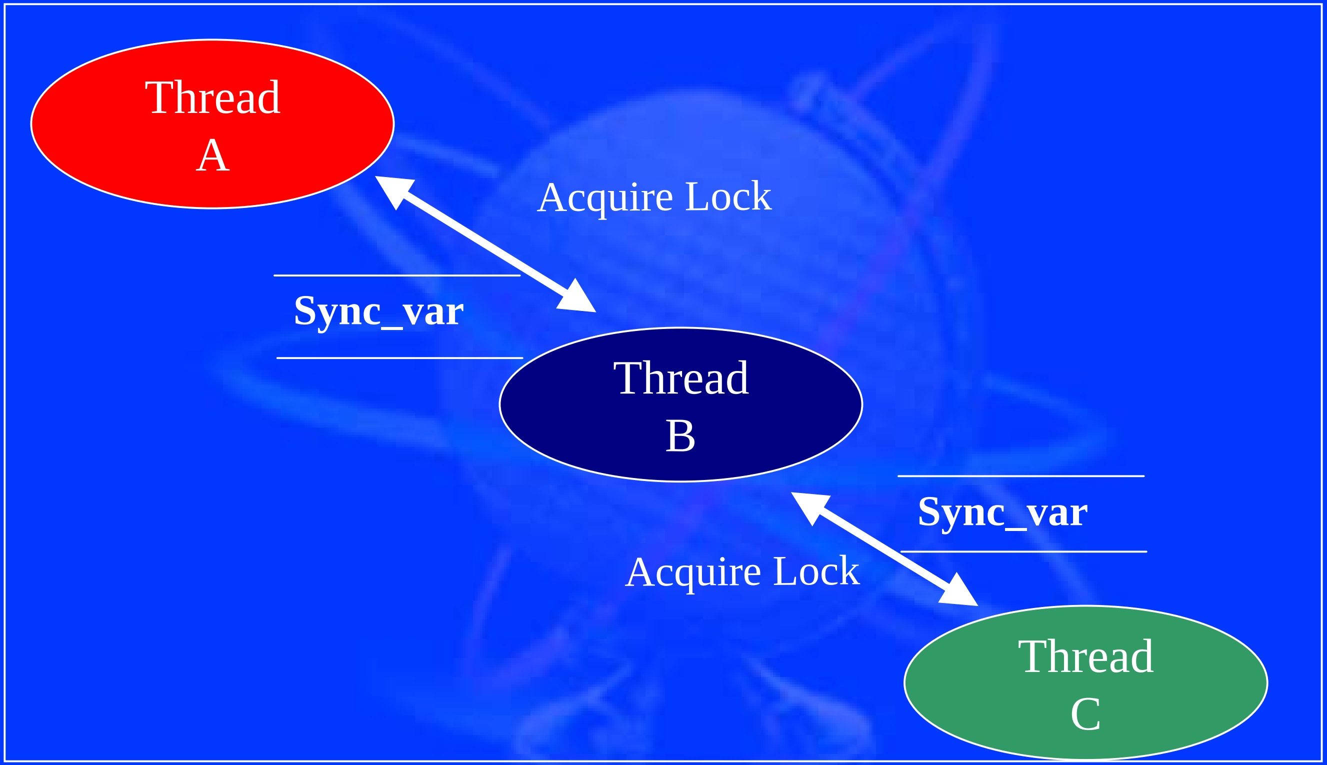
Halt Long Idle Thread

```
// Create thread pool and suspend them
Num_created = create_threadpool( num_threads);
for (jj = 0; jj < Num_created; jj ++) {
  ResumeThread(thread_handles[jj]);
}
for (jj = 0; jj < Num_created; jj ++) {
  // Call OS to free up resource for idle thread
  ret = WaitForMultipleObjects(Num_events,
  event_handles, FALSE, INFINITE) ;
  // check return code
}
```

Thread-blocking API can free up processor

Intel Developer Forum Fall 2001

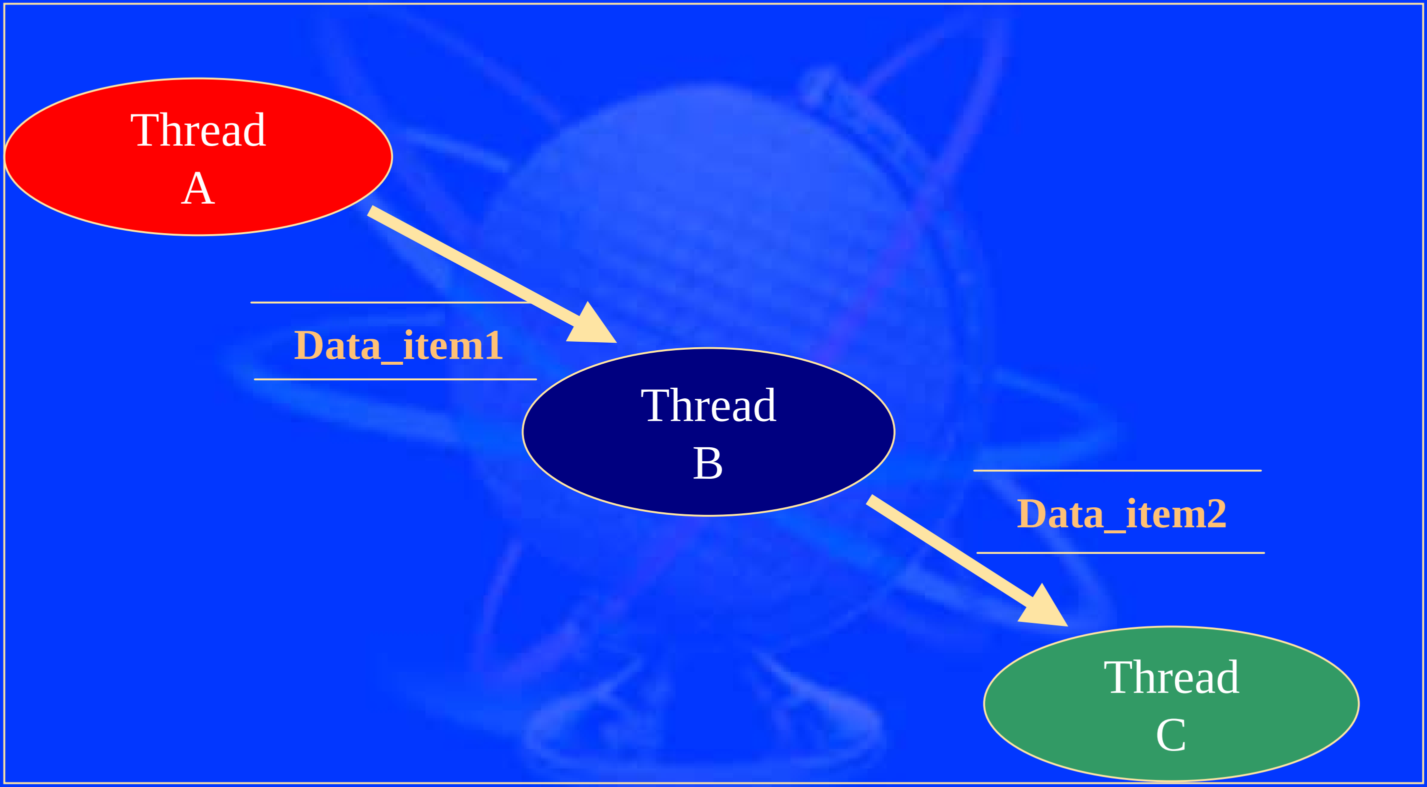
Congested Spin Locks



Excessive contentions create hot locks

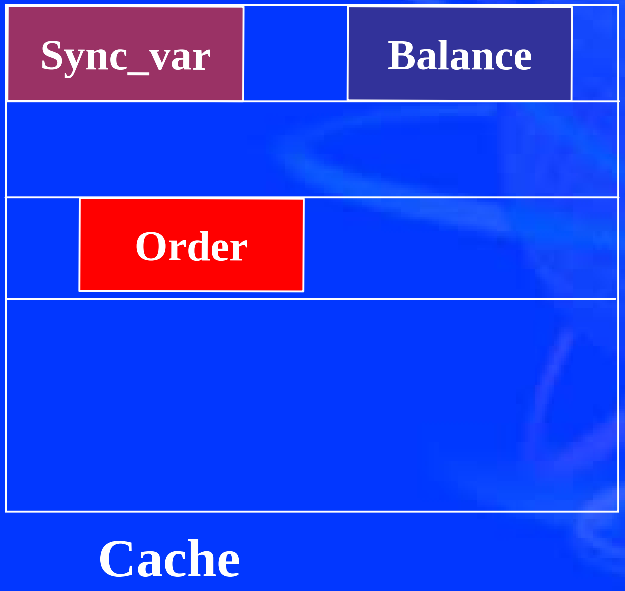
Intel Developer Forum Fall 2001

Pipelining Spin Locks

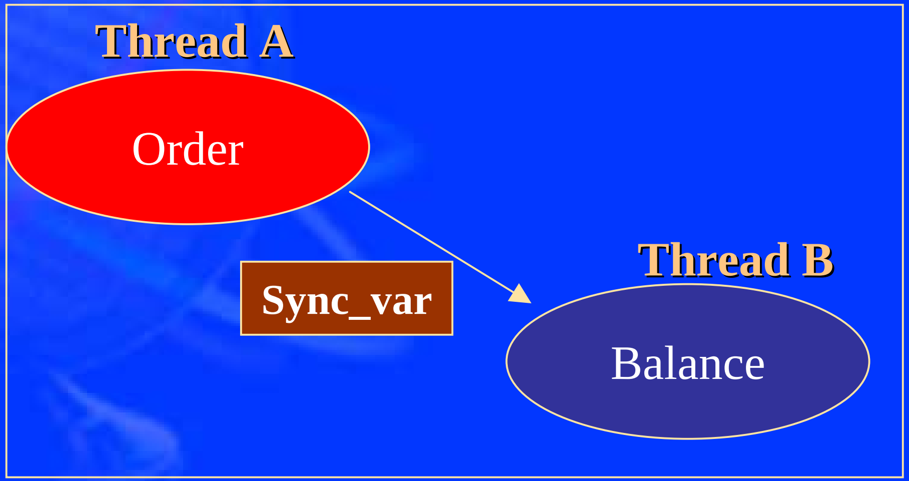


Pipelined spin locks reduce contentions

Falsely-Shared Cache Line

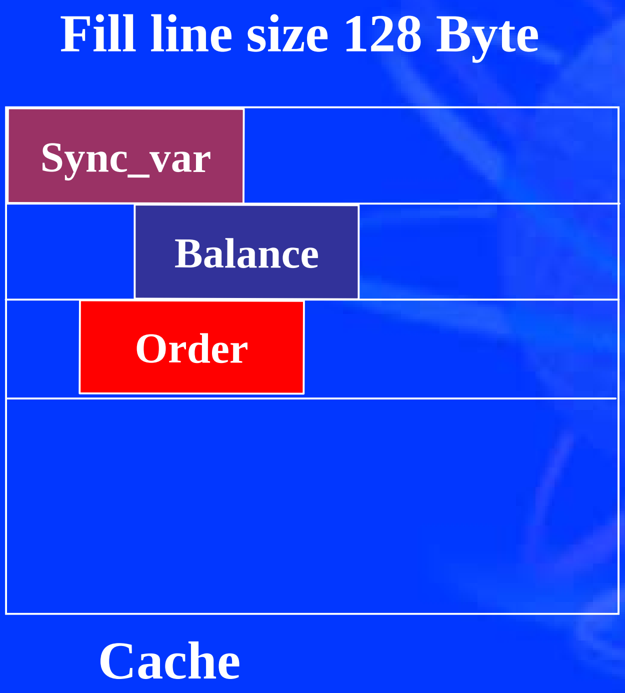


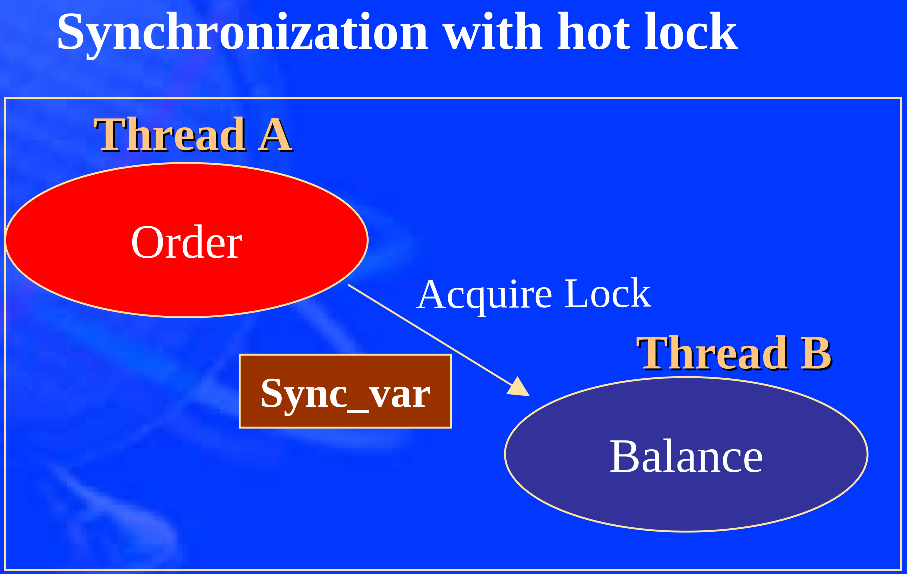
Synchronization without lock



The data “Balance” is falsely shared

Prevent Falsely-Shared Cache Line





Place sync_var on separate cache line

Examples

- Test speed-up of function threading
- Test speed-up of data threading
- Coarse-grain-threading kernels
  - Thread-M: Find maximum/minimum
  - Thread-A: Averages and Variances
  - Thread-S: Calculate Spline
  - Thread-H: Long-latency hazard

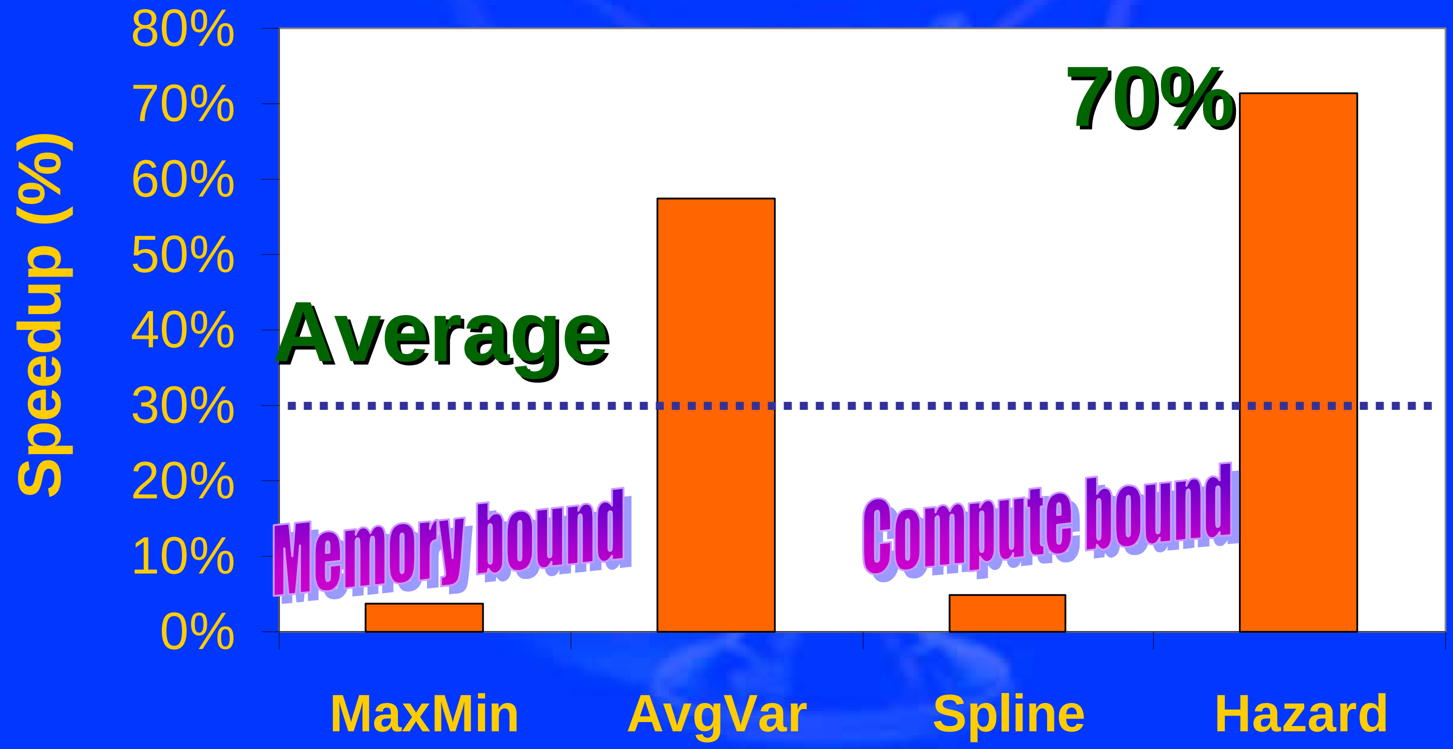
Performance Methodology

- Measure duration of fixed total work
- Single-threaded execution as baseline
  - No threading overhead
- Speed-up of 2 data threads
  - Include threading overhead
- Speed-up of 2 functional threads
  - Include threading overhead

Intel Developer Forum Fall 2001

Data-Domain Performance

Two Threads, Same Tasks

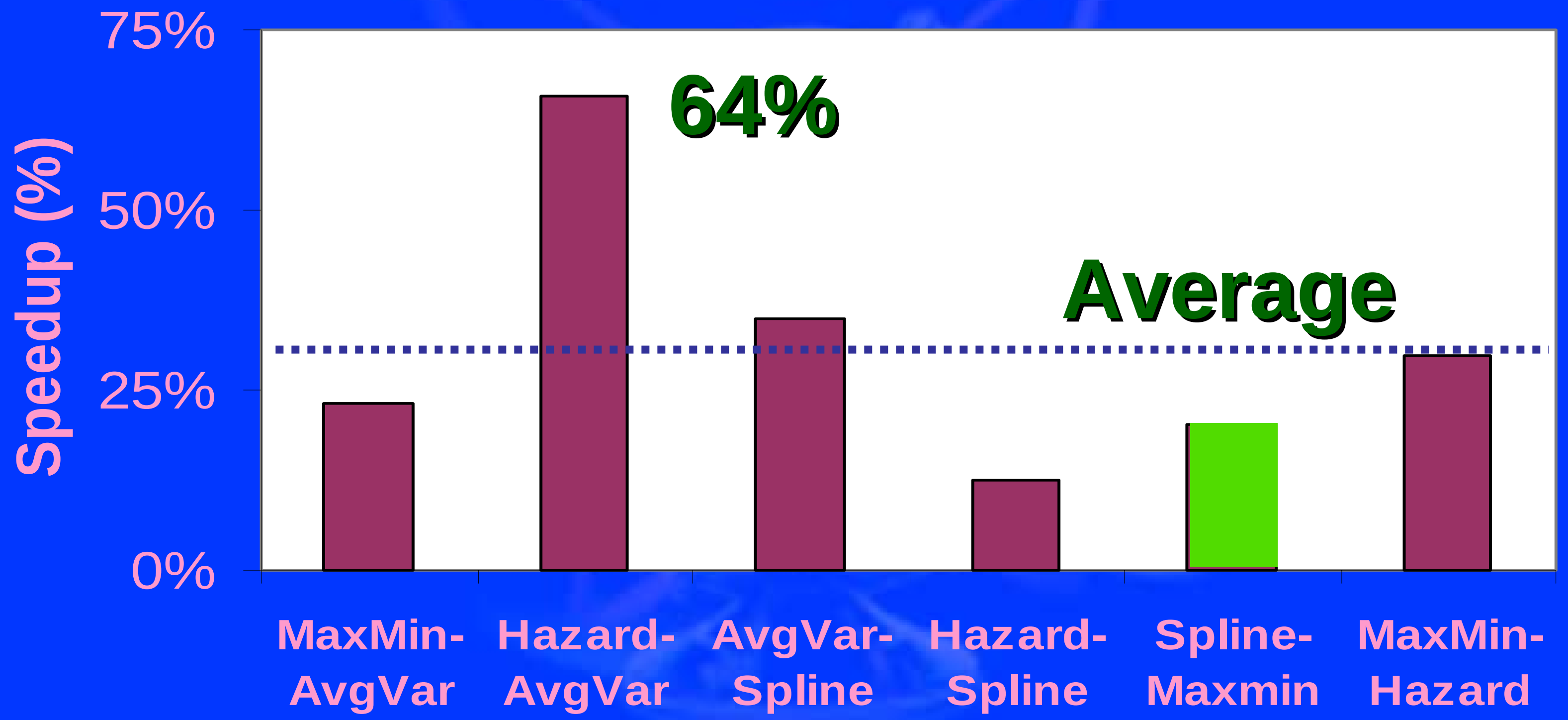


Hyper-Threading technology increases data threading performance

Intel Developer Forum Fall 2001

Function Domain Performance

Two Threads, Different Tasks



Hyper-Threading technology increases functional threading performance

Summary

- Hyper-Threading technology provides 2 logical processor in one physical package
- Workload parallelism pays and make thread synchronization painless
- Programming techniques to manage hardware resources are readily available
- Significant speedup of both data threads and functional threads due to Hyper-Threading technology

Call to Action

- Thread your applications take advantage of Hyper-Threading technology!

Visit:
http://developer.intel.com/technology/hyperthread

Collateral

- Web Sites
  - http://developer.intel.com/design/pentium4/applnots
  - http://developer.intel.com/design/pentium4/manuals
- Documentation and application notes
  - IA-32 Intel® Architecture Software Developer's Manual
  - Intel Pentium® 4 and Intel Xeon™ Processor Optimization Manual
  - Intel App Note AP485 - "Intel Processor Identification and CPU Instructions"
  - Intel App Note AP 949 " Using Spin-Loops on Intel Pentium 4 Processor and Intel Xeon Processor"
  - Intel App Note "Detecting Support for Hyper-Threading Technology Enabled Processors"

Collateral (Cont'd)

- Multi-processor technology
  - "Multiprocessing and Multithreading - Current Implementations at the Processor Level", Intel Corp.
- IDF Courses:
  - "Exploiting parallelism Using Intel Compiler", Software Tools and Optimization Track, IDF, Fall 2001
  - "Multithreaded Programming with OpenMP*", Workstation and Technical Computing Track, IDF, Fall 2001

Collateral (Cont'd)

- Multi-threaded programming
  - An Introduction to Programming with Threads, by Andrew Birrell; http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-035.html
  - Win32 API documentation is available on-line to MSDN subscribers at http://premium.microsoft.com/msdn/library
  - http://www.kai.com
  - Illinois-Intel Multithreading Library: Multithreading Support for Intel Architecture Based Multiprocessor Systems, Intel Technology Journal, 1998

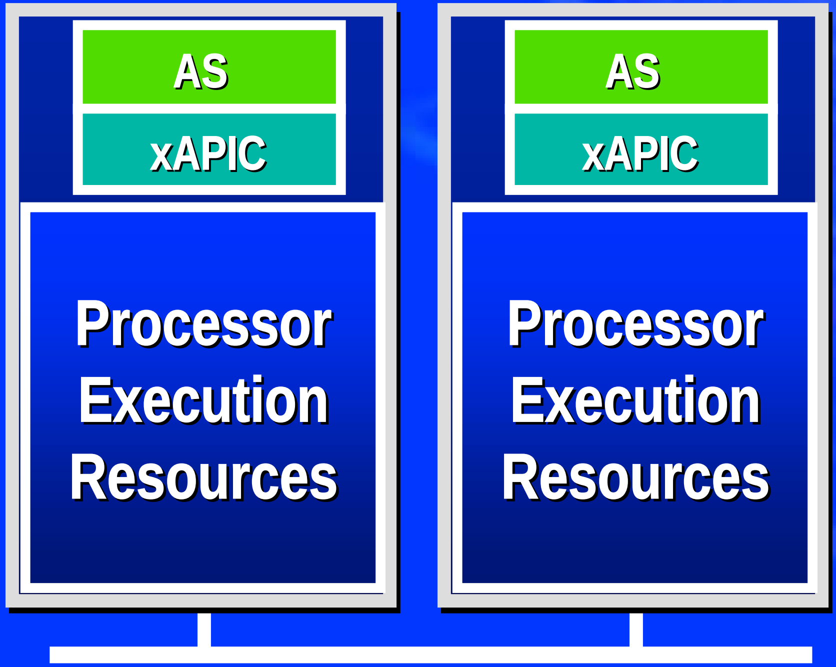
Collateral (Cont'd)

- Multi-threaded programming Tools
  - Intel C++ Compiler : http://developer.intel.com/software/products/compilers/
  - KAI C++ Compiler: http://www.kai.com
- Intel Developer Service
  - http://developer.intel.com/ids

Backup

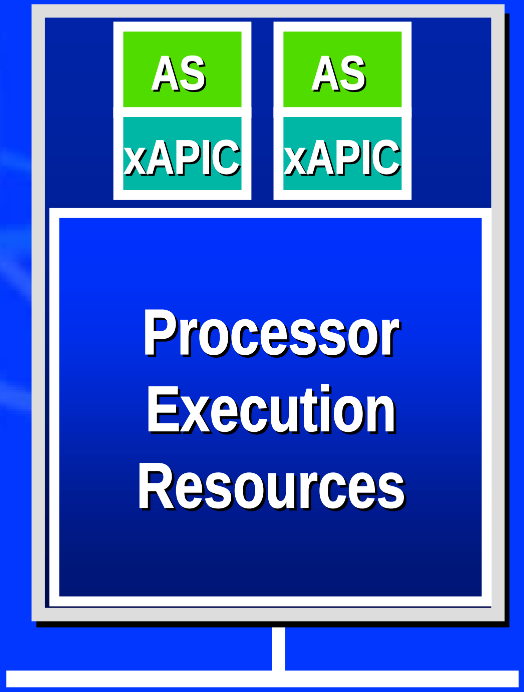
2 Logical Processors != 2 Physical Processors

MP w/ duplicated resources



System Bus

MP w/ shared resources



System Bus