

The Lecture Contains:

- ☰ Intel Compilers and Threading Tools
- ☰ Some Issues Involved in Parallel Programming
- ☰ Contd . . .
- ☰ We Need Help
- ☰ Intel Tools
- ☰ Build Correct Optimize Cluster
- ☰ Intel Compiler
- ☰ Example Code
- ☰ Intel C++ Compiler Output View
- ☰ Intel C++ Compiler Tips
- ☰ Intel Thread Checker
- ☰ Intel Thread Checker Output View
- ☰ Example Code

◀ Previous Next ▶

Intel Compilers and Threading Tools

Motivation

- The future is Multicores.

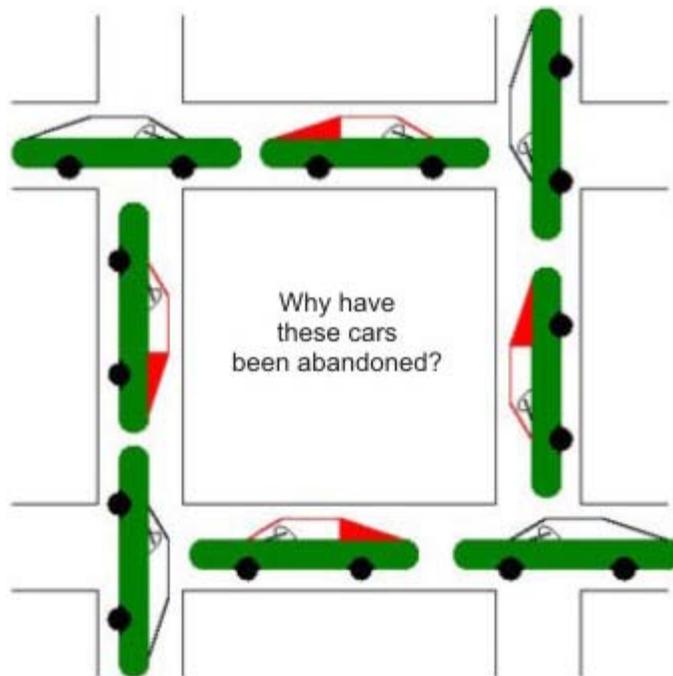


- Programmers must inevitably write parallel program constructs.
 - Complexity is way higher than sequential programming.
- What about existing sequential programs?
 - How to parallelize them?

◀ Previous Next ▶

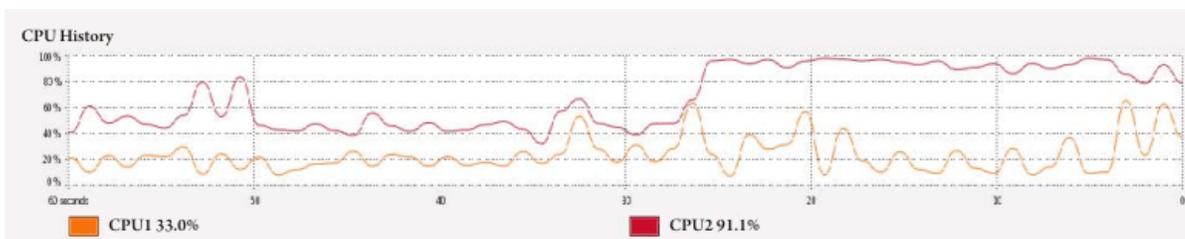
Some Issues Involved in Parallel Programming

- Deadlocks, Livelocks.



Contd . . .

- Data Races (Synchronization)
 - A race condition is a kind of error condition peculiar to parallel programs. The outcome of a program changes as the relative scheduling of units of execution (UEs) varies. This is often the result of modifications to a shared or global variable by more than one UE without an appropriate synchronization mechanism (such as a barrier, mutex, or semaphore).
- Load Balancing



◀ Previous Next ▶

Contd . . .

- Core Utilization
 - The application must try to utilize 100% of the units of execution available.
- Correctness. (Consistency models)
 - How do we decide if the output is correct or not.
 - Must follow some rules.
 - Sequential Consistency, Linearizability
- Lock Free and Wait Free
 - A method is said to be wait-free if it guarantees that every call finishes its execution in a finite number of steps.
 - A method is said to be lock-free if it guarantees that infinitely often some call finishes in a finite number of steps.

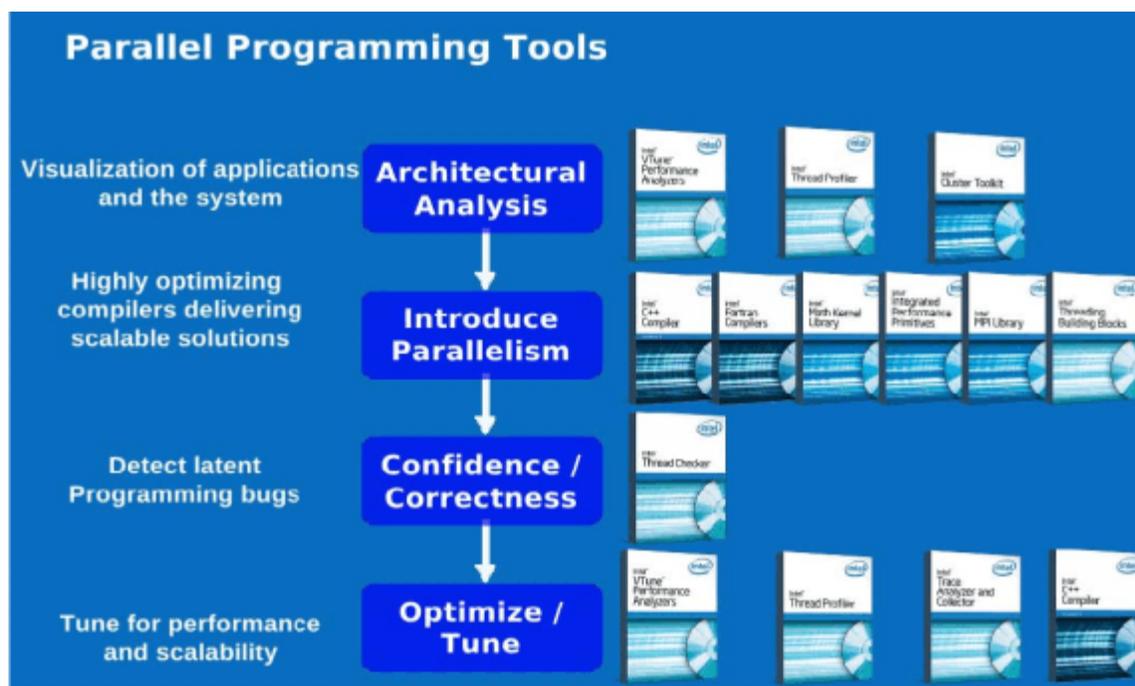
We Need Help

- Tools can help us finding out these issues and in some cases, resolve them too.
- Intel® Corporation has developed some tools for the development of multi-threaded applications under their Higher Performance Computing Category.



◀ Previous Next ▶

Intel Tools



Build Correct Optimize Cluster

- Build
 - Intel Compilers
- Correct
 - Intel Thread Checker
- Optimize
 - Intel Vtune Performance Analyzer
 - Intel Thread Proler
- Cluster
 - Intel Cluster Toolki

Intel Compiler

- Based on The Programming Language
 - Intel C++ Compiler
 - Intel Fortran Compiler



- Features
 - Multi-Threaded Application Support (OpenMP)
 - High Performance Parallel Optimizer (HPO)
 - Automatic Vectorization
 - Profile-Guided Optimization (PGO)
 - Interprocedural Optimization (IPO)

Example Code

prog1.cpp

```

1  using namespace std;
2
3  int main() {
4
5      int x[ 100000];
6      int a[ 100000], b[100000 ], c [100000]m;
7
8      for ( int i=0; i<100000; i+- {
9          for ( int j=0; j<100000; j++ {
10             x[ i ] = i;
11         }
12     }
13
14     for ( int i =0; i< 100000; i++) {
15         for ( int j =0; j< 1000; j++) {
16             a[ j ] =b [ j ] + c [ j ];
17         }
18     }
19 }

```

◀ Previous Next ▶

Intel C++ Compiler Output View

```

Shell - Konsole
Session Edit View Bookmarks Settings Help

bash-3.1$ /opt/intel/cc/19.0.023/bin/./icc -parallel -par-threshold:0 -par-report2 prog1.cpp
  procedure: main
prog1.cpp(8): (col. 2) remark: LOOP WAS AUTO-PARALLELIZED.
prog1.cpp(9): (col. 3) remark: loop was not parallelized: existence of parallel dependence.
prog1.cpp(14): (col. 2) remark: loop was not parallelized: existence of parallel dependence.
prog1.cpp(15): (col. 3) remark: LOOP WAS AUTO-PARALLELIZED.
bash-3.1$ /opt/intel/cc/19.0.023/bin/./icc -parallel -par-threshold:100 -par-report2 prog1.cpp
  procedure: main
prog1.cpp(8): (col. 2) remark: LOOP WAS AUTO-PARALLELIZED.
prog1.cpp(9): (col. 3) remark: loop was not parallelized: existence of parallel dependence.
prog1.cpp(14): (col. 2) remark: loop was not parallelized: existence of parallel dependence.
prog1.cpp(15): (col. 3) remark: loop was not parallelized: insufficient computational work.
bash-3.1$ █

```

Intel C++ Compiler Tips

- The default compilation is with O2 (level-2 optimization) flag set.
- The compiler autoperallelizes only if O2/O3 flag is set.
- Remember to set -par-report2/3 flag while autoperallelization for a complete diagnosis report.
- It is better to use O0 flag for beginners of OpenMP because some optimizations(code motion) can produce unexpected results.
- Refer to man pages when in doubt.

◀ Previous Next ▶

Intel Thread Checker

- Features

- Intel Thread Checker detects data races, deadlocks, stalls, and other threading issues. It can detect the potential for these errors even if the error does not occur during an analysis session.
- Pinpoint the function, context, line, variable, and call stack in the source code to aid analysis and repair of bugs.
- Identify nearly impossible-to-nd data races and deadlocks using an advanced error detection engine. Helps to reduce untraceable errors.
- Errors do not need to actually occur to be detected. Make the code as more robust.

Intel Thread Checker Output View

Relation...	ID	Short Description	Severity	Descriptor	Count	F
1	1	Write -> Read data-race		Memory read at "openmp1.cpp":15 conflicts with a prior memory write at "openmp1.cpp":16 (flow dependence)	9632	False
1	2	Write -> Read data-race		Memory read at "openmp1.cpp":15 conflicts with a prior memory write at "openmp1.cpp":15 (flow dependence)	9711	False
1	3	Write -> Write data-race		Memory write at "openmp1.cpp":15 conflicts with a prior memory write at "openmp1.cpp":15 (output dependence)	9711	False
1	4	Write -> Read data-race		Memory read at "openmp1.cpp":16 conflicts with a prior memory write at "openmp1.cpp":16 (flow dependence)	9632	False
1	5	Write -> Read data-race		Memory read at "openmp1.cpp":16 conflicts with a prior memory write at "openmp1.cpp":12 (flow dependence)	127302	False
1	6	Write -> Write data-race		Memory write at "openmp1.cpp":16 conflicts with a prior memory write at "openmp1.cpp":16 (output dependence)	9632	False
1	7	Write -> Write data-race		Memory write at "openmp1.cpp":12 conflicts with a prior memory write at "openmp1.cpp":12 (output dependence)	239	False
1	8	Read -> Write data-race		Memory write at "openmp1.cpp":16 conflicts with a prior memory read at "openmp1.cpp":16 (anti dependence)	79	False
1	9	Write -> Write data-race		Memory write at "openmp1.cpp":16 conflicts with a prior memory write at "openmp1.cpp":16 (output dependence)	79	False
1	10	Read -> Write data-race		Memory write at "openmp1.cpp":16 conflicts with a prior memory read at "openmp1.cpp":16 (anti dependence)	79	False

Severity distribution

Legend:

- Unclassified
- Remark
- Information
- Caution
- Warning
- Error
- Filtered

Diagnostic: Stack Traces Source View

◀ Previous Next ▶

Example Code

```
#include <pthread.h>
int Count;

void *findCount()
{
    Count++; //Flow, Anti, Output Dependency
}
int main()
{
    int i, rc;
    pthread_t h[2];

    Count = 0;

    for ( i = 0; i < 2; ++i)
    {
        rc = pthread_create (&h[i], 0, findCount, NULL);
    }

    for ( i = 0; i < 2; ++i)
    {
        rc = pthread_join (h[i], 0);
    }
    return 0;
}
```

◀ Previous Next ▶