Automatic Code Parallelization

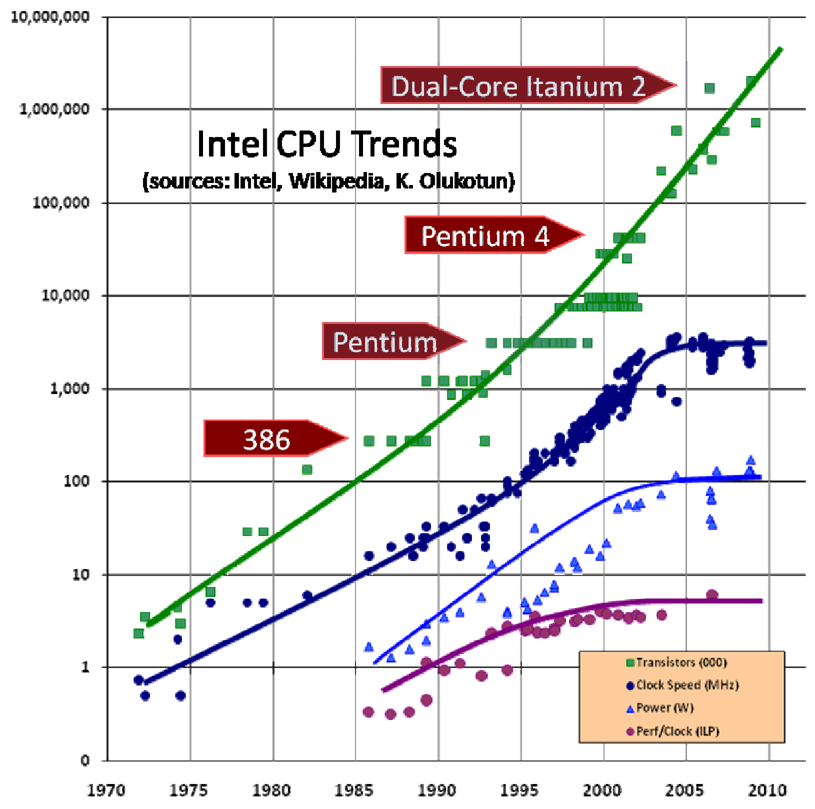
SAPIENZA
UNIVERSITÀ DI ROMA

Alessandro Pellegrini

PhD program in Computer Science and Engineering
Cycle XXVI

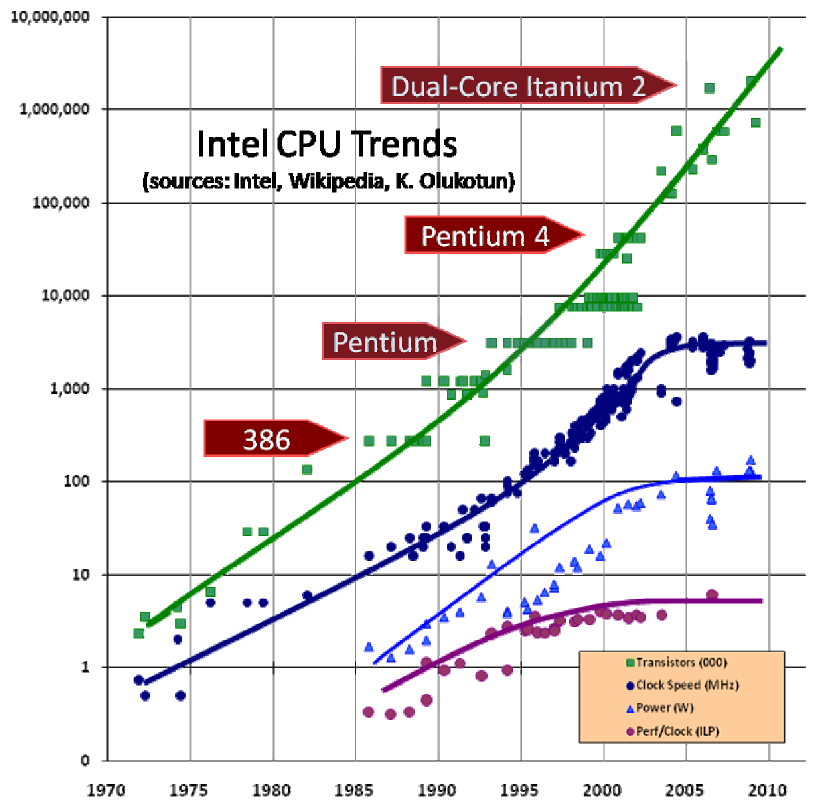
Motivations: Technological Trend

- Moore's Law changed since year 2003.



Motivations: Technological Trend

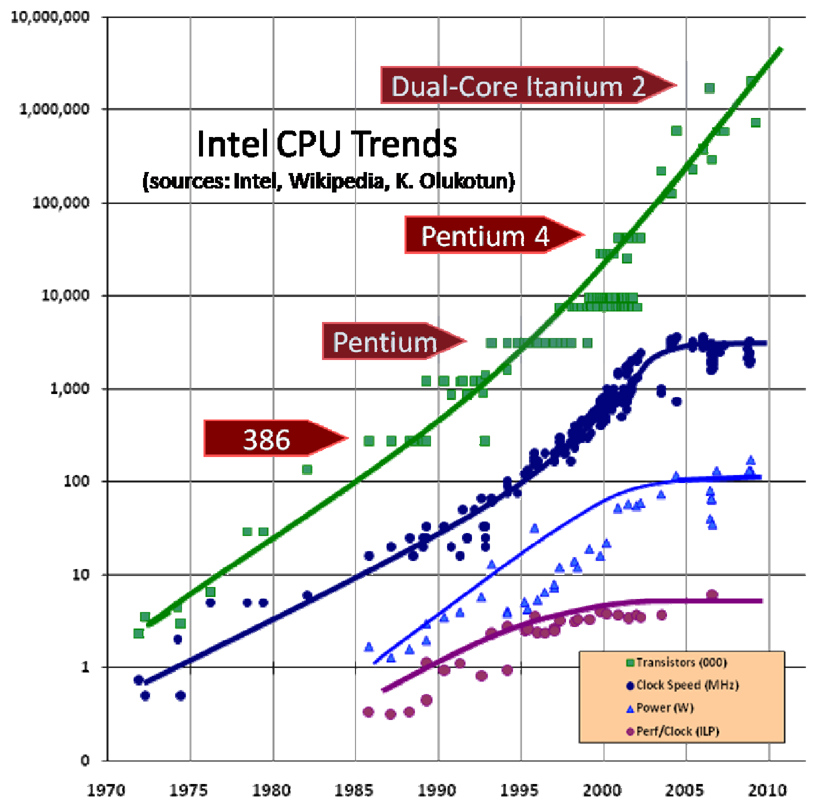
- Moore's Law changed since year 2003
- 130W is considered an upper bound.



Motivations: Technological Trend

- Moore's Law changed since year 2003
- 130W is considered an upper bound.

$$P = CV^2 f$$



Your Free Lunch

"There ain't no such thing as a free lunch"

— R.A. Heinlein, *The Moon is a Harsh Mistress*

- Our free lunch is already over
- What can we do about it?
- What *are* we going to do about it?

Your Free Lunch

"There ain't no such thing as a free lunch"

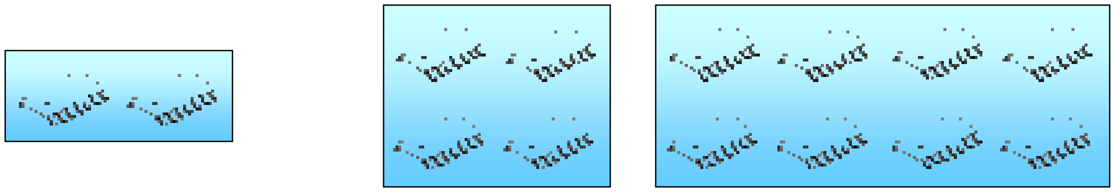
— R.A. Heinlein, *The Moon is a Harsh Mistress*

- Our free lunch is already over
- What can we do about it?
- What are we going to do about it?

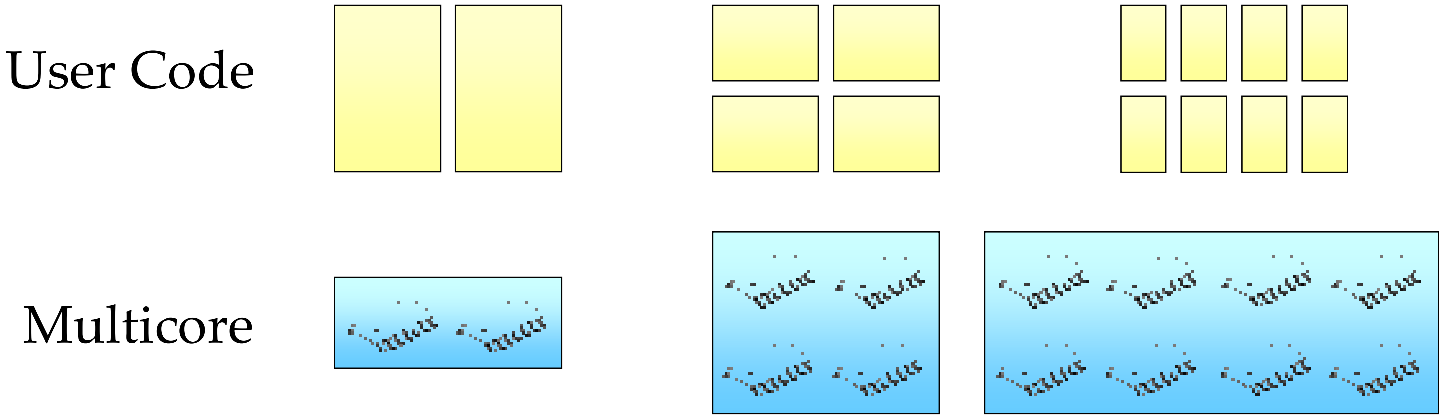
Future is Parallel

Multicore Software Scaling

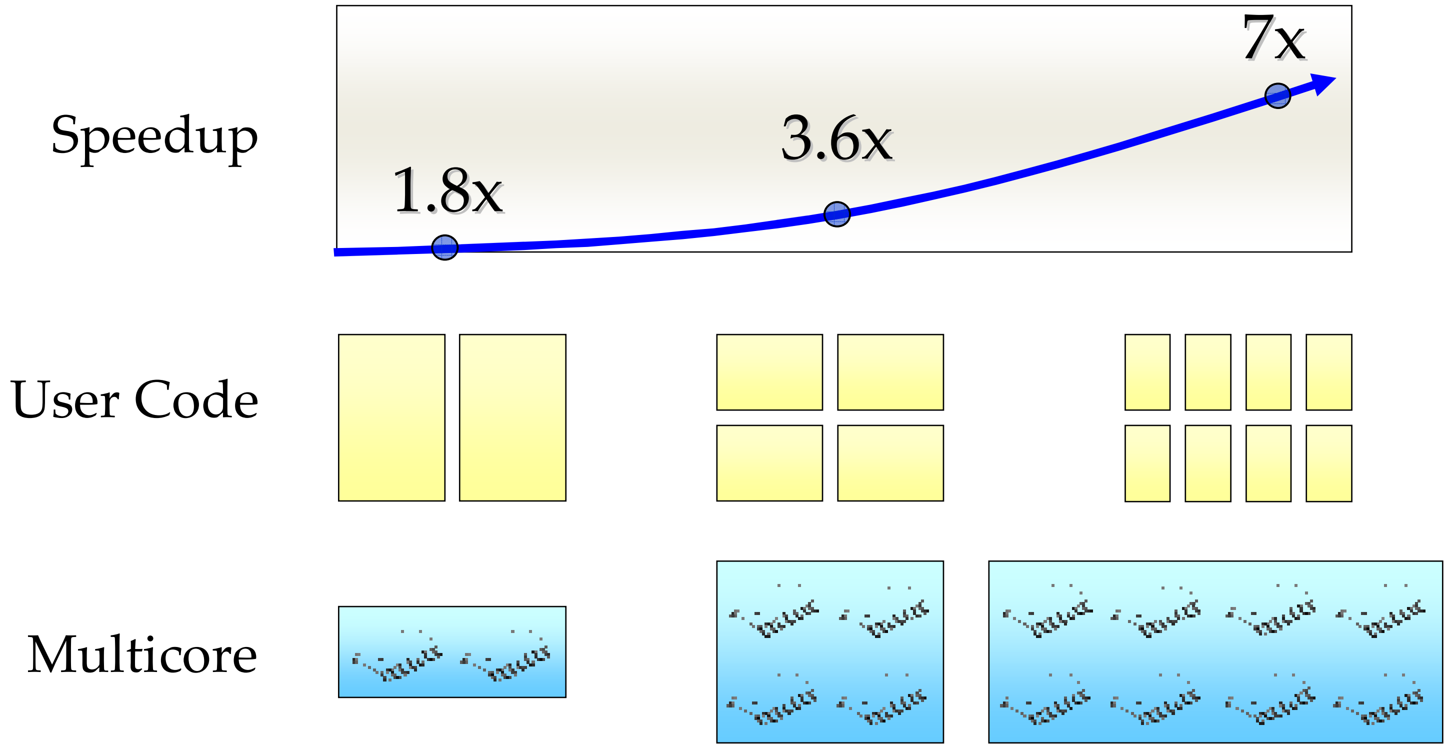
Multicore



Multicore Software Scaling



Multicore Software Scaling

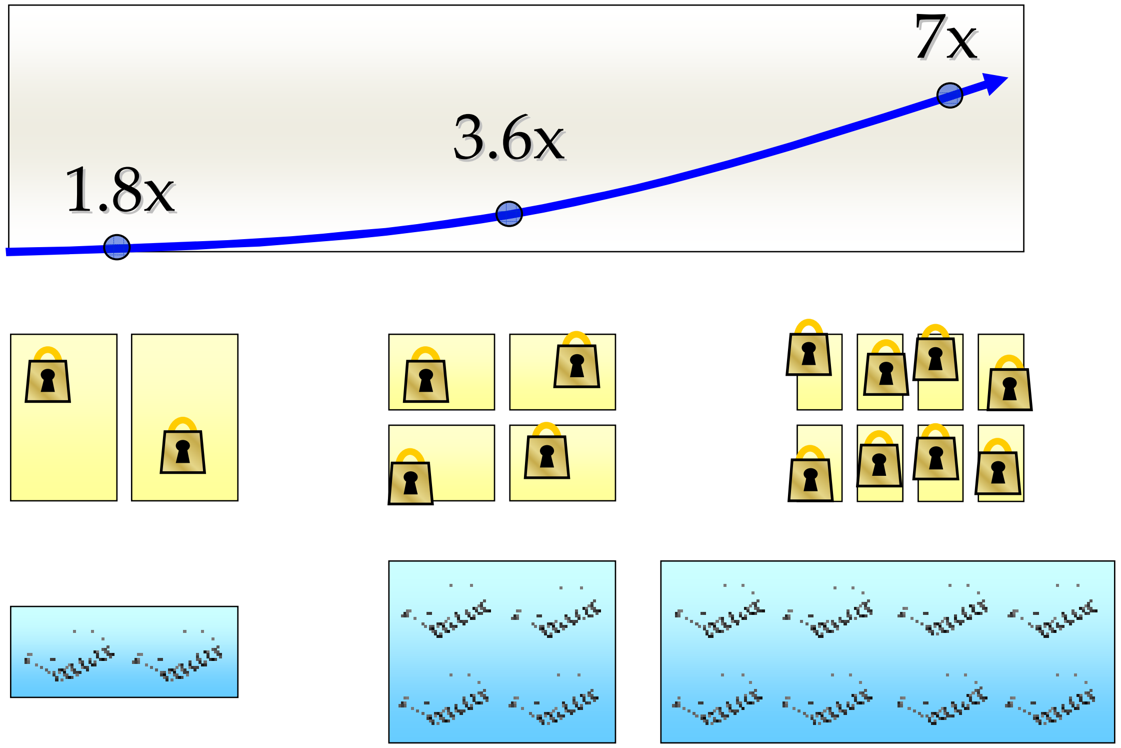


Multicore Software Scaling

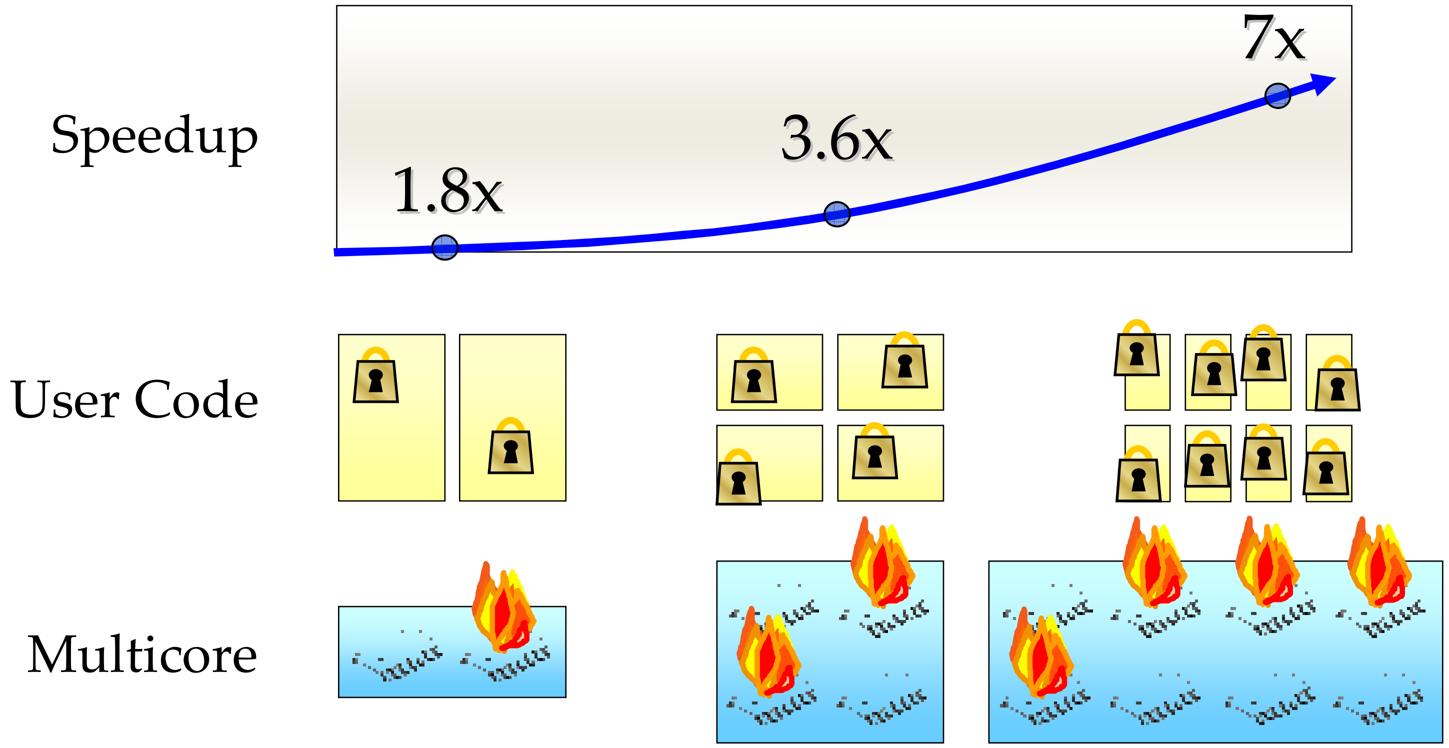
Speedup

User Code

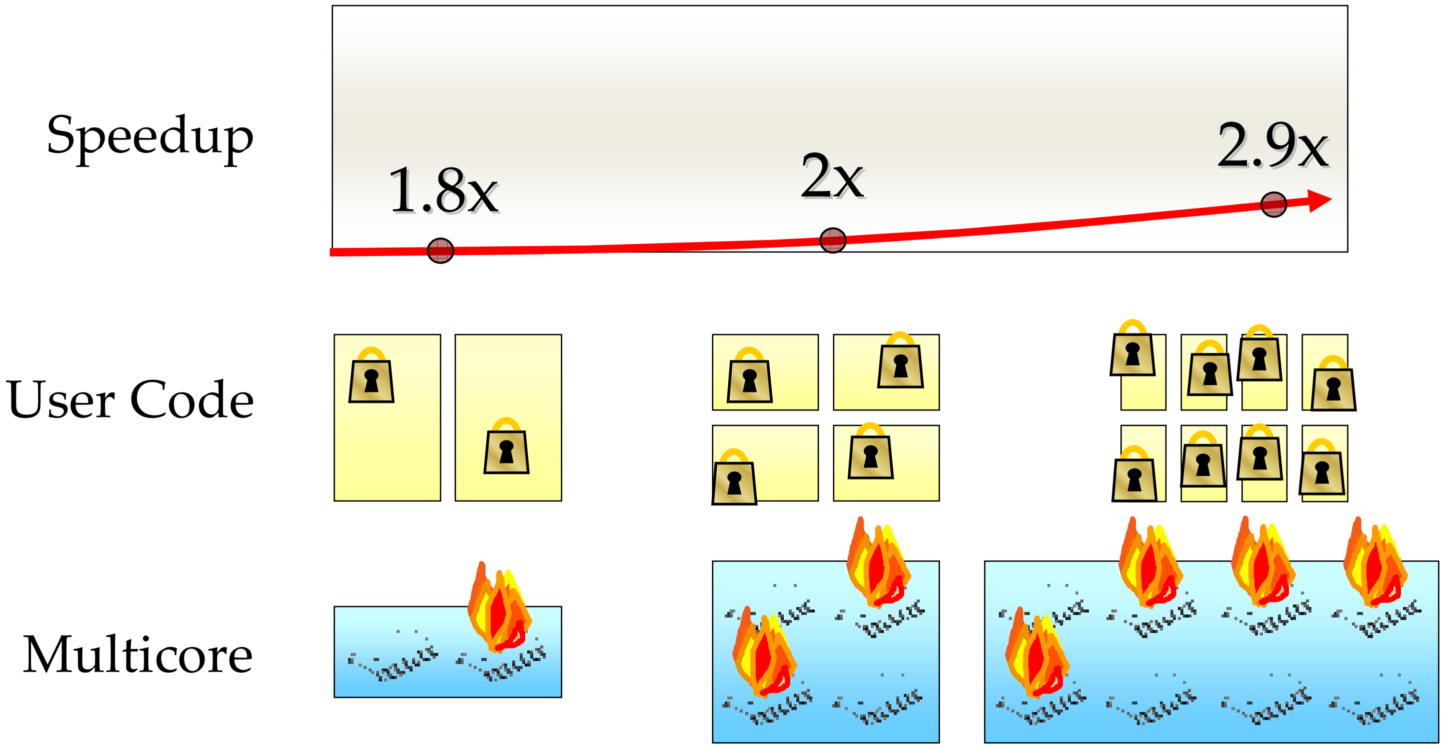
Multicore



Multicore Software Scaling



Multicore Software Scaling



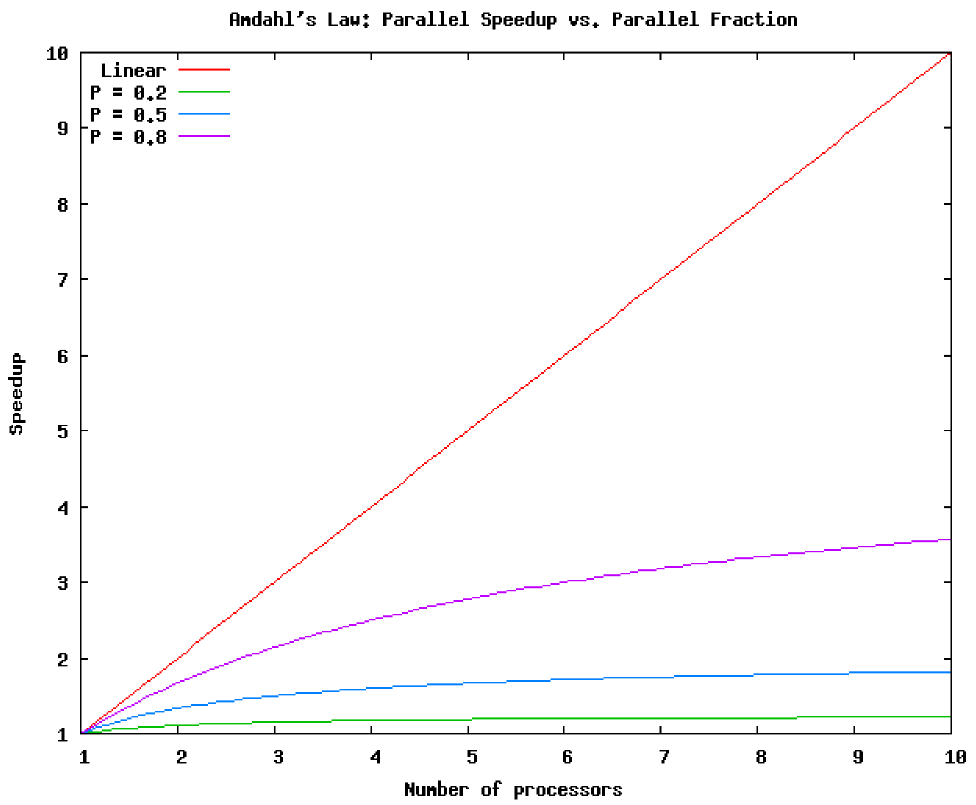
Theoretical Bound

Amdahl's Law

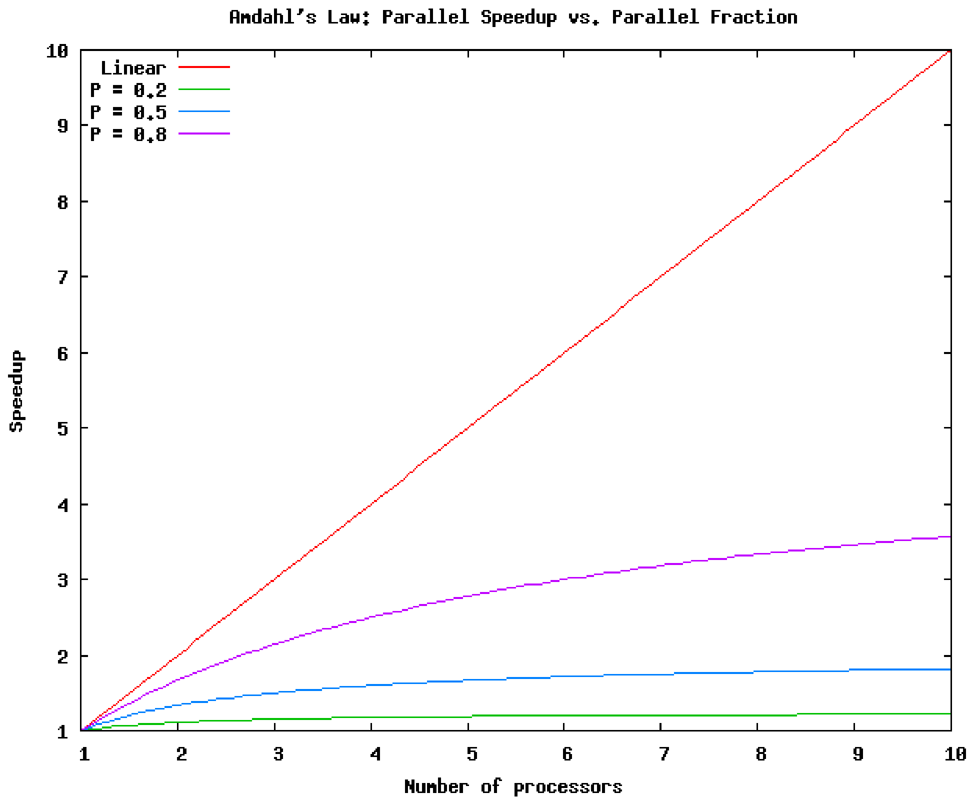
Every program has some fraction of its code (P) which can be run in parallel. In the optimal case, given that we have S processing units, the maximum achievable speedup for the entire program is:

$$S_{\max} = \frac{1}{(1-P) + \frac{P}{S}}$$

Theoretical Bound



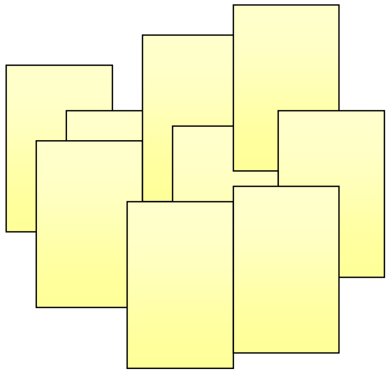
Theoretical Bound



If parallel fraction is 80% of the whole program:

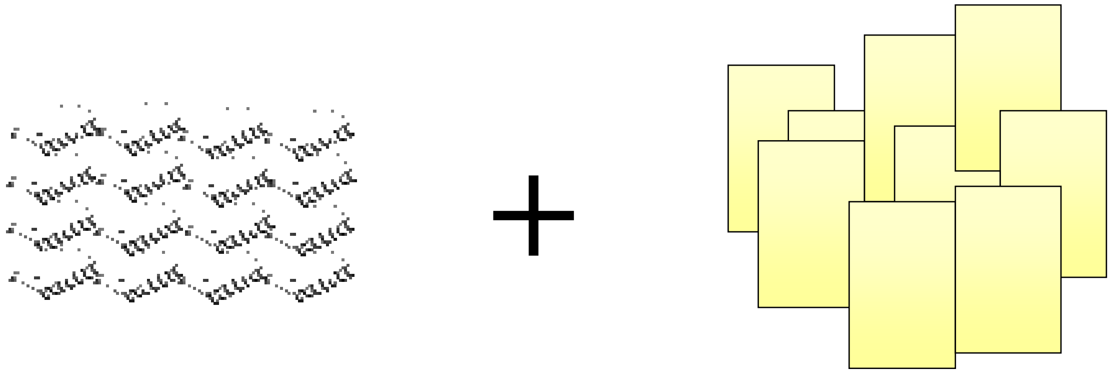
$$\lim_{S \to +\infty} S_{\max} = \frac{1}{1 - 0.8} = 5$$

The Needs



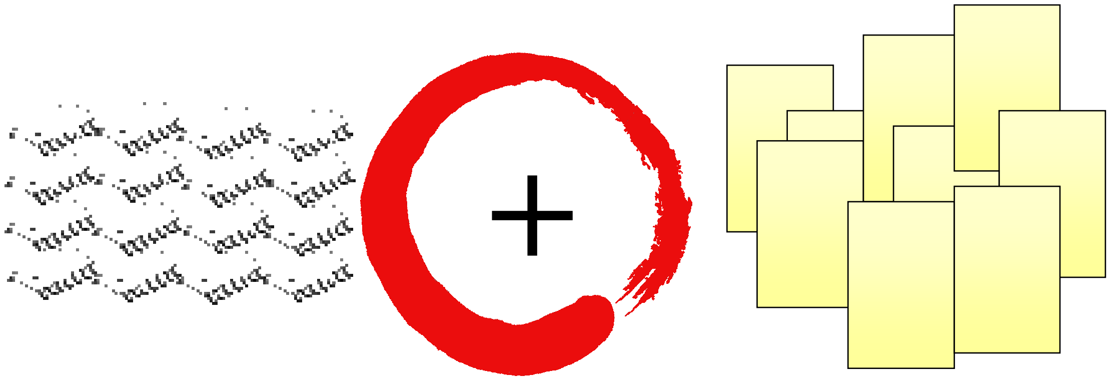
- Multicore / Clusters are a consolidated reality
- Both Scientific & Commodity Users
- Parallel Programming has been a privilege of few experts so far

The Needs



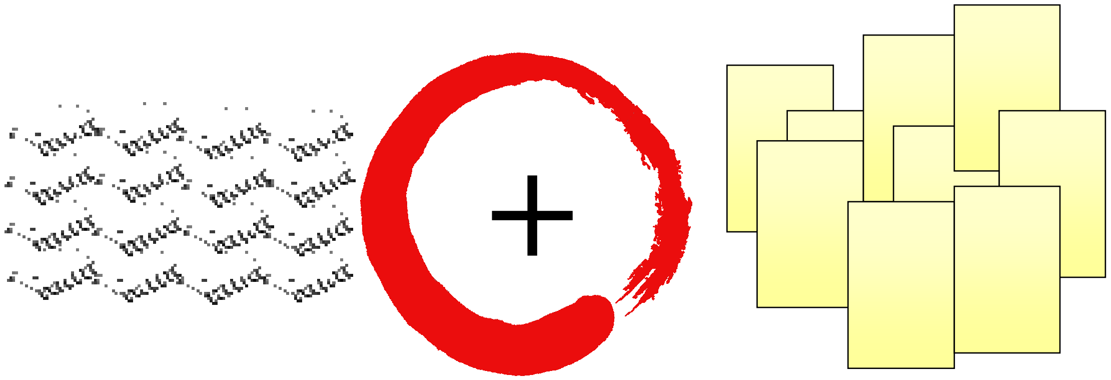
- Multicore / Clusters are a consolidated reality
- Both Scientific & Commodity Users
- Parallel Programming has been a privilege of few experts so far

The Needs



- Multicore / Clusters are a consolidated reality
- Both Scientific & Commodity Users
- Parallel Programming has been a privilege of few experts so far

The Needs



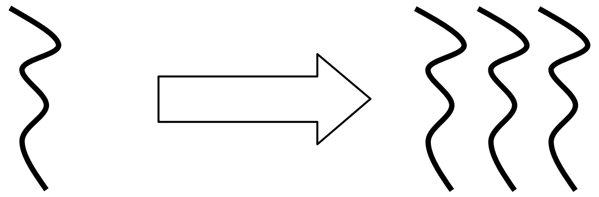
As parallel processors become ubiquitous, the key question is:

How do we convert the applications we care about into parallel programs?

The Goals

- Produce tools and methodologies aimed at supporting *automatic program parallelization*.
- **Transparency**: allowing programmers to fully exploit all the semantic power of languages, automatically performing operations they should know nothing about
- **Efficiency**: make the parallel programs run fast
- Trade-off between the two.

State of the Art



- Is parallelism expressed *explicitly* or not?
- How do the individual processes interact?
- How do they communicate data and synchronize with each other?

State of the Art

Models of Sharing Communication

- Shared Memory:
  - one-sided
- Message Passing:
  - More cumbersome: two-sided protocol & marshalling

Explicit Parallelism

State of the Art

- Transactional Memories:

  a generic non-blocking synchronization construct, based on *transactions*.

State of the Art

- Event-Driven Programming:

  the flow of the program is determined by user-/software-generated events or messages from other threads/processes.

State of the Art

- Parallelizing Compilers:

  try to identify which are the instruction blocks which can be run in parallel, and try to optimize the code for parallel execution relying on runtime libraries

State of the Art

Parallel Languages & Specifications

- Communicating Sequential Processes (CSP): a formal language which is able to express interaction patterns between components
- OpenMP: a specification for an annotation system and runtime libraries

State of the Art

- Speculative Processing:
  - A means to improve parallel-programs performance wrt serial fractions
  - Tries to guess the outcome of parallel parts and executes serial ones before knowing the actual results
  - Typically based on a retry-until-commit approach

Main Issues & Solution Proposals

- Strong trade-off between transparency and efficiency
- Non general-purpose solutions

- Extend the semantic comprehension of programs
- Low level: Machine Code & Assembly

Methodologies to be Investigated

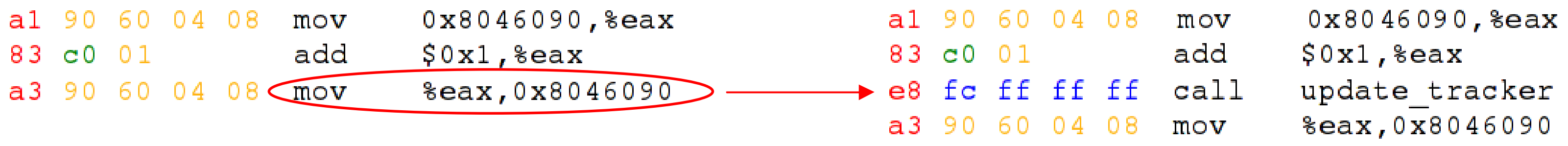
- Code Instrumentation and Mangling
- Software Multiversioning
- Mutagen Code
- Speculativity:
  - Execution Flow Prediction
  - Interprocedural Semantic Analysis
  - Process Cloning
  - Work Unit definition & Worker Threads
  - Log/Restore Facilities

Achieved Results

In the PDES scenario:

x86 (32-/64-bits) Code Instrumentor [IC1]

- Disassembly caching for reduced runtime overhead



```
a1 90 60 04 08  mov    0x8046090,%eax
83 c0 01        add    $0x1,%eax
a3 90 60 04 08  mov    %eax,0x8046090
```

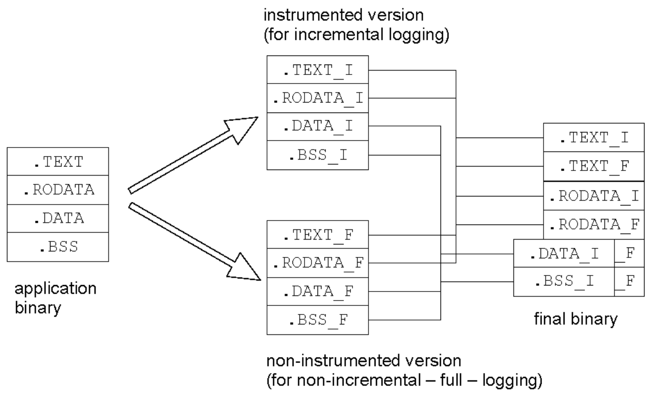
```
a1 90 60 04 08  mov    0x8046090,%eax
83 c0 01        add    $0x1,%eax
e8 fc ff ff ff  call   update_tracker
a3 90 60 04 08  mov    %eax,0x8046090
```

Achieved Results

In the PDES scenario:

Code Instrumentor: *Multiversioning* [IC3]

- Fast switching between versions
- Stability of the optimal performance



Achieved Results

In the PDES scenario:

Optimistic Simulator (ROOT-Sim) has been publicly released [IC5]

http://www.dis.uniroma1.it/~hpdcs/ROOT-Sim

A feedback from the community!

Current Activities

- Rule-programmable Instrumentor:
  - Easily address different scenarios
- Programmatic Speculative Stack Frame Analysis:
  - Runtime support for tracking execution patterns
- New Memory Allocator:
  - Enhance Data Separation

Publications

[IC5]	Alessandro Pellegrini, Roberto Vitali and Francesco Quaglia *The Rome Optimistic Simulator: Core Internals and Programming Model* Proc. Of the 4-th International ICST Conference of Simulation Tools and Techniques (SIMUTOOLS)
[IC4]	Alessandro Pellegrini, Roberto Vitali and Francesco Quaglia *An Evolutionary Algorithm to Optimize Log/Restore Operations within Optimistic Simulation Platforms* Proc. Of the 4-th International ICST Conference of Simulation Tools and Techniques (SIMUTOOLS)
[IC3]	Roberto Vitali, Alessandro Pellegrini and Francesco Quaglia Proc. Of the 18-th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)
[IC2]	Roberto Vitali, Alessandro Pellegrini and Francesco Quaglia *Benchmarking Memory Management Capabilities within ROOT-Sim* Proc. of the 13-th IEEE/ACM International Symposium on Distributed Simulations and Real-Time Applications (DS-RT)
[IC1]	Alessandro Pellegrini, Roberto Vitali and Francesco Quaglia *Di-DyMeLoR: Logging only Dirty Chunks for Efficient Management of Dynamic Memory Based Optimistic Simulation Objects* Proc. of the 23-rd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simiulation (PADS)

Thank You

Questions?