

---

# Long-Term Research in High End Computing

## The PITAC Report and Its Implications for the Petaflops Initiative

Ken Kennedy

PITAC Co-Chair

Petaflops Keynote

<http://www.cs.rice.edu/~ken/Presentations/Petaflops.pdf>

# Presentation Outline

---

- About PITAC and the Report
  - Charter and Methodology
  - Findings and Recommendations
    - Emphasis on High End
- Implications for Petaflops
  - Funding implications
  - Challenges
    - Architecture, Applications, Software
- Memory Hierarchy
  - Regular techniques, irregular applications
- Conclusions

# PITAC Charter

---

- The Committee shall provide an independent assessment of:
  - Progress made in implementing the High-Performance Computing and Communications (HPCC) Program;
  - Progress in designing and implementing the Next Generation Internet initiative;
  - The need to revise the HPCC Program;
  - Balance among components of the HPCC Program;
  - Whether the research and development undertaken pursuant to the HPCC Program is helping to maintain United States leadership in advanced computing and communications technologies and their applications;
  - Other issues as specified by the Director of the Office of Science and Technology.
    - Review of the entire IT investment strategy — is it meeting the nation's needs

# PITAC Membership

---

- **Co-Chairs:**

- Bill Joy, Sun Microsystems

- Ken Kennedy, Rice

- **Members:**

- Eric Benhamou, 3Com

- Ching-chih Chen, Simmons

- Steve Dorfman, Hughes

- Bob Ewald, SGI

- Sherri Fuller, U of Washington

- Susan Graham, UC Berkeley

- Danny Hillis, Disney, Inc

- John Miller, Montana State

- Raj Reddy, Carnegie Mellon

- Larry Smarr, UIUC

- Les Vadasz, Intel

- Steve Wallach, Centerpoint

- Vinton Cerf, MCI

- David Cooper, LLNL

- David Dorman, PointCast

- David Farber, Penn

- Hector Garcia-Molina, Stanford

- Jim Gray, Microsoft

- Robert Kahn, CNRI

- David Nagel, AT&T

- Ted Shortliffe, Stanford

- Joe Thompson, Miss. State

- Andy Viterbi, Qualcomm

- Irving Wladawsky-Berger, IBM

# Activities

---

- Evaluation of Federal Research Investment Portfolio
  - Plans reviewed for each of the major areas:
    - High End Computing and Computation
    - Large Scale Networking
    - Human Centered Computer Systems
    - High Confidence Systems
    - Education, Training, and Human Resources

# Activities

---

- Evaluation of Federal Research Investment Portfolio
  - Plans reviewed for each of the major areas:
    - High End Computing and Computation
    - Large Scale Networking
    - Human Centered Computer Systems
    - High Confidence Systems
    - Education, Training, and Human Resources
- Review of Balance in Federal Research Portfolio
  - Fundamental versus Applied
    - Based on our own definition of these terms
  - High versus Low Risk
  - Long versus Short-Term

# Principal Finding

---

- **Drift Away from Long-Term Fundamental Research**
  - Agencies pressed by the growth of IT needs
    - IT R&D budgets have grown steadily but not dramatically
    - IT industry has accounted for over 30 percent of the real GDP growth over the past five years, but gets only 1 out of 75 Federal R&D dollars
    - Problems solved by IT are critical to the nation—engineering design, health and medicine, defense
  - Most IT R&D agencies are mission-oriented
    - Natural and correct to favor the short-term needs of the mission
- **This Trend Must Be Reversed**
  - Continue the flow of ideas to fuel the information economy and society

# Remedy

---

- Double the Federal IT R&D Investment to 2 billion dollars per year
  - Ramp up over five years
  - Focus on increasing fundamental research
- Invest in Key Areas Needing Attention
  - Software
  - Scalable Information Infrastructure
  - High-End Computing
  - Social, Economic, and Workforce Issues
- Develop a Coherent Management Strategy
  - Diversify modes of support

# Software

---

- Findings:
  - Demand for software far exceeds the nation's ability to produce it
  - The nation depends on fragile software
  - Technologies to build reliable and secure software are inadequate
  - The nation is under-investing in fundamental software research
- Recommendations:
  - Fund more fundamental research in software development methods and component technologies
    - Sponsor a national library of software components
  - Support fundamental research in human-computer interfaces and interaction
  - Make fundamental software research an absolute priority
    - Make software research a substantive component of every major IT research initiative

# Scalable Information Infrastructure

---

- Findings:

- The Internet has grown well beyond the intent of its original designers
- Our nation's dependence on the information infrastructure is increasing daily
- We cannot safely extend what we currently know to more complex systems
- Learning how to build large-scale, highly reliable and secure systems requires research

- Recommendations:

- Increase funding in research and development of core software and communications technologies aimed directly at the challenge of scaling the information infrastructure
- Expand the Next Generation Internet testbeds to include additional industry partnerships in order to foster the rapid commercialization and deployment of enabling technologies

# High-End Computing

---

- Findings:

- High-end computing is essential for science and engineering research
- High-end computing is an enabling element of the United States national security program
- New applications of high-end computing are ripe for exploration
- Suppliers of high-end systems suffer from unusual market pressures
  - High-end market not large—must depend on commercially viable components
  - Scalable parallel architectures not ideal for every application
  - Specialized software tools needed to overcome the usability gap
  - Resources to produce software tools are not commensurate with the needs

# High-End Recommendations

---

- Research:
  - Fund research into innovative computing technologies and architectures
  - Fund R&D on software for improving the performance of high-end computing

# High-End Recommendations

---

- Research:
  - Fund research into innovative computing technologies and architectures
  - Fund R&D on software for improving the performance of high-end computing
  - Drive high-end computing research by trying to attain a sustained petaops/petaflops on real applications by 2010 through a balance of hardware and software strategies

# High-End Recommendations

---

- Research:

- Fund research into innovative computing technologies and architectures
- Fund R&D on software for improving the performance of high-end computing
- Drive high-end computing research by trying to attain a sustained petaops/petaflops on real applications by 2010 through a balance of hardware and software strategies



# High-End Recommendations

---

- **Research:**
  - Fund research into innovative computing technologies and architectures
  - Fund R&D on software for improving the performance of high-end computing
  - Drive high-end computing research by trying to attain a sustained petaops/petaflops on real applications by 2010 through a balance of hardware and software strategies
- **Facilities**
  - Fund the acquisition of the most powerful high-end computing systems to support science and engineering research

# High-End Recommendations

---

- **Research:**
  - Fund research into innovative computing technologies and architectures
  - Fund R&D on software for improving the performance of high-end computing
  - Drive high-end computing research by trying to attain a sustained petaops/petaflops on real applications by 2010 through a balance of hardware and software strategies
- **Facilities**
  - Fund the acquisition of the most powerful high-end computing systems to support science and engineering research
- **Management**
  - Expand the federal High-End Computing and Computation (HECC) program to include all of the major elements of the government investments in high-end computing

# Social, Economic, Workforce Issues

---

- **Recommendations:**
  - Expand federal research on the social and economic impacts of information technology diffusion and adoption
  - Expand Federal initiatives and government, university, industry partnerships to increase IT literacy and ensure equitable access
  - Develop programs to help address the shortage of high-technology workers
    - Increase research funding to help grow faculty
    - Develop new educational programs to re-train information technology workers whose skills have become outdated
    - Encourage increased participation by women and minorities
    - Increase the annual cap on H-1B visas as a short-term remedy to address the shortage of skilled IT workers

# Modes of Support

---

- Finding:
  - The Federal IT R&D funding profile is incomplete
- Recommendations:
  - Diversify the modes of research support to foster projects of broader scope and longer duration
    - Teams, funding for 3 years or more
  - Fund virtual centers for Expeditions into the 21st Century
    - Virtual “think tanks” focused on revolutionary IT
    - Academia, government, and industry scientists live in the technology future and investigate a unique IT focus
  - Establish a program of Enabling Technology Centers
    - Centers of excellence in computer science and engineering (similar to NSF STCs)
    - Applications of information and communications technology

# Management

---

- Recommendations:
  - Designate NSF as the lead Federal agency to coordinate fundamental information technology research
  - Expand the current coordination mechanisms already in place
    - Currently used for HPCC and NGI
    - Agency representatives should have budget authority
  - Establish a comprehensive annual review of research programs
    - Oversight by a Presidential Advisory Committee

# Questions

---

- Can we increase long-term research by rebudgeting?
  - No, because the short-term work addresses essential problems

# Questions

---

- Can we increase long-term research by rebudgeting?
  - No, because the short-term work addresses essential problems
- Why doesn't industry fund this?
  - Innovation based on venture capital
  - Thin margins

# Questions

---

- Can we increase long-term research by rebudgeting?
  - No, because the short-term work addresses essential problems
- Why doesn't industry fund this?
  - Innovation based on venture capital
  - Thin margins
- Can the research community absorb another billion \$ per year?
  - Yes: \$400M in unused capacity, \$350M in facilities, \$250M in expanded capacity (2500 new researchers over 5 years)

# Questions

---

- Can we increase long-term research by rebudgeting?
  - No, because the short-term work addresses essential problems
- Why doesn't industry fund this?
  - Innovation based on venture capital
  - Thin margins
- Can the research community absorb another billion \$ per year?
  - Yes: \$400M in unused capacity, \$350M in facilities, \$250M in expanded capacity (2500 new researchers over 5 years)
- What is the right balance between research and facilities?
  - Our guideline: < 25 percent of the increase in any given year should go to facilities

# Questions

---

- Can we increase long-term research by rebudgeting?
  - No, because the short-term work addresses essential problems
- Why doesn't industry fund this?
  - Innovation based on venture capital
  - Thin margins
- Can the research community absorb another billion \$ per year?
  - Yes: \$400M in unused capacity, \$350M in facilities, \$250M in expanded capacity (2500 new researchers over 5 years)
- What is the right balance between research and facilities?
  - Our guideline: < 25 percent of the increase in any given year should go to facilities
- Is NSF the right agency to lead in coordination?
  - Its mission is fundamental research, but is it too conservative?

# Good News

---

- **Administration Budget**
  - **Additional \$366 million in FY 2000**
    - NSF: \$146 million, with \$35 million for facilities
    - DoD: \$100 million, with \$70 million for DARPA
    - DOE: \$70 million for SSI
    - NASA: \$38 million
    - NOAA: \$6 million
    - NIH: \$6 million
  - **Prospects for successive years unclear**
- **Congress**
  - **Briefings have begun**
    - **Reception positive (so far)**

# Implications for Petaflops

---

- There will be increased research funding
  - Petaflops/petaops should drive research in high end computing
    - Architecture and technology
    - Software
  - Resources to support facilities
    - DOE and NSF
  - Resources to support application development
    - Goal: Drive software and architecture research

# Implications for Petaflops

---

- There will be increased research funding
  - Petaflops/petaops should drive research in high end computing
    - Architecture and technology
    - Software
  - Resources to support facilities
    - DOE and NSF
  - Resources to support application development
    - Goal: Drive software and architecture research
- Software will be essential
  - Compilers and Tools
  - Libraries
  - Programming Systems and Environments

# Petaflops Architectures

---

- Petaflops/Petaops by 2007

# Petaflops Architectures

---

- Petaflops/Petaops by 2007
- Faster processors, high processor count
  - projection for 2007:
    - 10,000 to 100,000 processors
    - 10-100 GF per processor

# Petaflops Architectures

---

- Petaflops/Petaops by 2007
- Faster processors, high processor count
  - projection for 2007:
    - 10,000 to 100,000 processors
    - 10-100 GF per processor
- Deep memory hierarchy
  - up to 10 levels

# Petaflops Architectures

---

- Petaflops/Petaops by 2007
- Faster processors, high processor count
  - projection for 2007:
    - 10,000 to 100,000 processors
    - 10-100 GF per processor
- Deep memory hierarchy
  - up to 10 levels
- High levels of parallelism required for performance
  - at least 10 million way parallelism must be found in the application
    - used to exploit parallel processors and hide memory latency

# Petaflops Architectures

---

- Petaflops/Petaops by 2007
- Faster processors, high processor count
  - projection for 2007:
    - 10,000 to 100,000 processors
    - 10-100 GF per processor
- Deep memory hierarchy
  - up to 10 levels
- High levels of parallelism required for performance
  - at least 10 million way parallelism must be found in the application
    - used to exploit parallel processors and hide memory latency
- Heterogeneous computational and network components
  - elements of Grid programming

# Future Applications

---

- **Application Complexity Increasing**
  - Irregular and adaptive computation
  - Multidisciplinary simulation and design
  - Multiscale simulations
  - Commercial applications
    - Java
  - Data intensive computation

# Future Applications

---

- **Application Complexity Increasing**
  - Irregular and adaptive computation
  - Multidisciplinary simulation and design
  - Multiscale simulations
  - Commercial applications
    - Java
  - Data intensive computation
- **Fewer Programmers → Application Composition**
  - Applications will involve many programs
  - MADIC study
    - 10,000 applications, untrusting developers
  - Requirements: language interoperability, composition tools

# HPC Software Successes

---

- **Compiler Memory Hierarchy Management**
  - register allocation, register and cache blocking, cache prefetching
  - Memory reorganization for parallelism
    - reduction of false sharing

# HPC Software Successes

---

- **Compiler Memory Hierarchy Management**
  - register allocation, register and cache blocking, cache prefetching
  - Memory reorganization for parallelism
- **Compiler Extraction of Parallelism**
  - Automatic parallelization
    - effective for loops on shared-memory multiprocessors
  - Language and compiler support for data parallelism
    - HPF available on every parallel platform

# HPC Software Successes

---

- **Compiler Memory Hierarchy Management**
  - register allocation, register and cache blocking, cache prefetching
  - Memory reorganization for parallelism
- **Compiler Extraction of Parallelism**
  - Automatic parallelization
  - Language and compiler support for data parallelism
- **Support for Portable Parallel Programming**
  - HPF, MPI, HPC++, OpenMP, Java

# HPC Software Successes

---

- **Compiler Memory Hierarchy Management**
  - register allocation, register and cache blocking, cache prefetching
  - Memory reorganization for parallelism
- **Compiler Extraction of Parallelism**
  - Automatic parallelization
  - Language and compiler support for data parallelism
- **Support for Portable Parallel Programming**
  - HPF, MPI, HPC++, OpenMP, Java
- **Parallel Libraries**
  - Communication, Math, Data Structures

# HPC Software Successes

---

- **Compiler Memory Hierarchy Management**
  - register allocation, register and cache blocking, cache prefetching
  - Memory reorganization for parallelism
- **Compiler Extraction of Parallelism**
  - Automatic parallelization
  - Language and compiler support for data parallelism
- **Support for Portable Parallel Programming**
  - HPF, MPI, HPC++, OpenMP, Java
- **Parallel Libraries**
  - Communication, Math, Data Structures
- **Integrated Tools**
  - Performance Analysis and Tuning, Debugging

# What We Must Do

---

- Find more parallelism
  - User specification
  - Compiler enhancement

# What We Must Do

---

- Find more parallelism
  - User specification
  - Compiler enhancement
- Ameliorate the memory hierarchy problem
  - Many levels of hierarchy
    - including I/O

# What We Must Do

---

- Find more parallelism
  - User specification
  - Compiler enhancement
- Ameliorate the memory hierarchy problem
  - Many levels of hierarchy
    - including I/O
- Keep the level of programming abstraction high
  - User focus on algorithm
  - System packages resources
  - Mechanisms for low-level control of performance
    - without abandoning all abstraction
  - Mechanisms for managing program change
    - minimal recovery time

# Memory Hierarchy Management

---

- Computation Reorganization
  - register and cache blocking
  - loop splitting

# Memory Hierarchy Management

---

- **Computation Reorganization**
  - register and cache blocking
  - loop splitting
- **Software Prefetching**
  - prefetch selection and placement

# Memory Hierarchy Management

---

- Computation Reorganization
  - register and cache blocking
  - loop splitting
- Software Prefetching
  - prefetch selection and placement
- Data Reorganization
  - variable grouping on cache lines
  - array storage reorganization
  - dynamic reorganization schemes

# Memory Hierarchy Management

---

- Computation Reorganization
  - register and cache blocking
  - loop splitting
- Software Prefetching
  - prefetch selection and placement
- Data Reorganization
  - variable grouping on cache lines
  - array storage reorganization
  - dynamic reorganization schemes
- Inclusion of I/O in Memory Hierarchy
  - extension of cache techniques
    - reorganization, prefetching

# Blocking in the Abstract

---

```
DO I = 1, N
  DO J = 1, N
    F(I) = F(I) + f(A(I), A(J))
  ENDDO
ENDDO
```

# Blocking in the Abstract

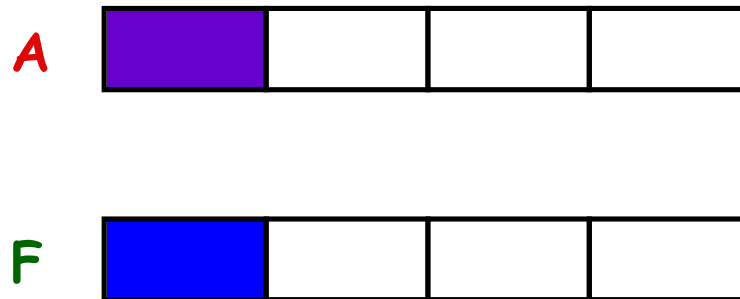
---

```
DO I = 1, N
  DO J = 1, N
    F(I) = F(I) + f(A(I), A(J))
  ENDDO
ENDDO

DO b1 = block1, blockN
  load F(block b1), A(block b1)
  DO b2 = block1, blockN
    if (b2 ≠ b1) load A(block b2)
    FOR I in b1
      FOR J in b2
        F(I) = F(I) + f(A(I), A(J))
      ENDFOR
    ENDFOR
  ENDDO
ENDDO
```

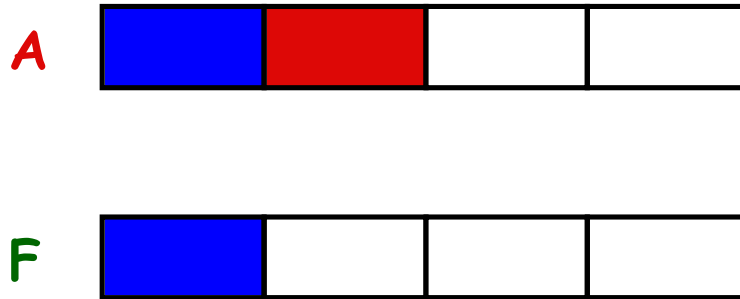
# Blocking Illustration

---



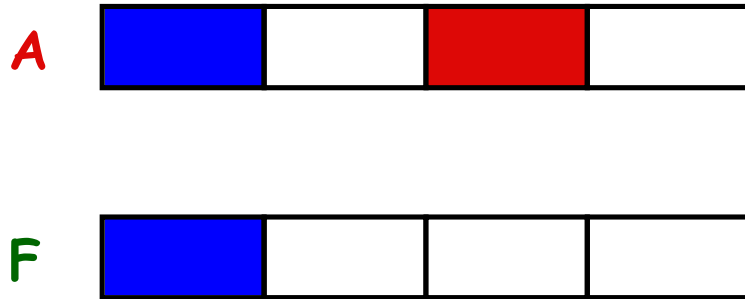
# Blocking Illustration

---



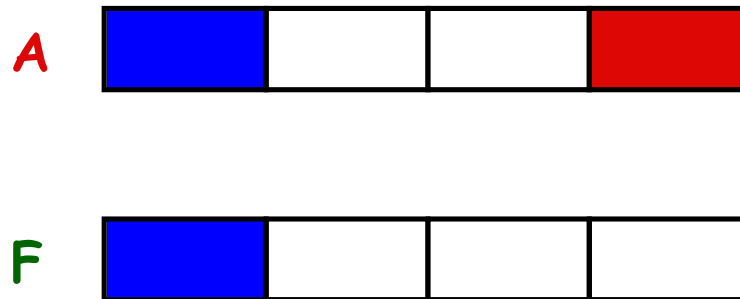
# Blocking Illustration

---



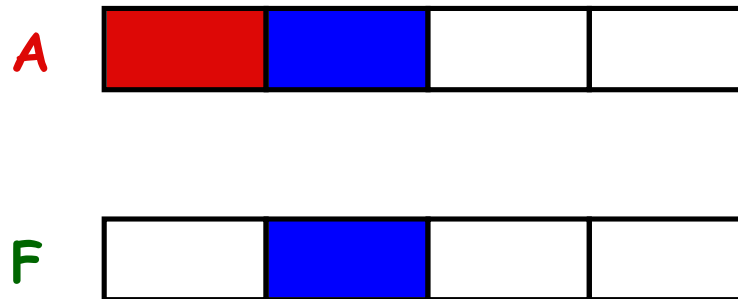
# Blocking Illustration

---



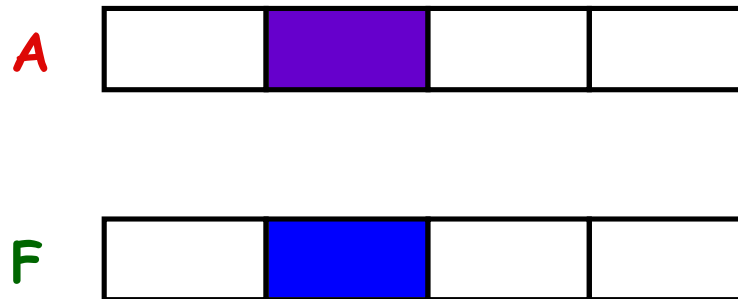
# Blocking Illustration

---



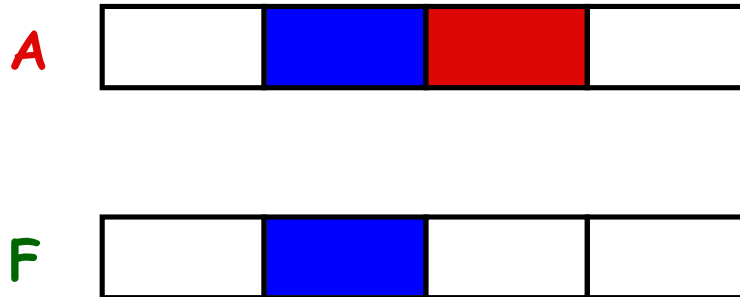
# Blocking Illustration

---



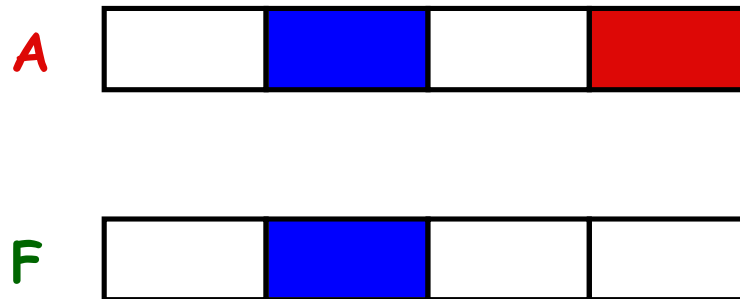
# Blocking Illustration

---



# Blocking Illustration

---



# Multilevel Blocking

---

```
DO I = 1, N  
  DO J = 1, M
```

$$F(I) = F(I) + f(A(I), A(J))$$

```
  ENDDO  
ENDDO
```

# Multilevel Blocking

---

```
DO I = 1, N, B1
  DO J = 1, M, B1
    DO ii = I, I+B1-1
      DO jj = J, J+B1-1
```

Blocking for L2

$F(ii) = F(ii) + f(A(ii), A(jj))$

```
      ENDDO
```

```
    ENDDO
```

```
  ENDDO
```

```
ENDDO
```

# Multilevel Blocking

---

```
DO I = 1, N, B1
  DO J = 1, M, B1
    DO I2 = I, I+B1-1, B1
      DO J2 = J, J+B1-1, B1
        DO ii = I2, I2+B2-1
          DO jj = J2, J2+B2-1
            F(ii) = F(ii) + f(A(ii), A(jj))
          ENDDO
        ENDDO
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

Blocking for L2

Blocking for L1

# Limitations of These Techniques

---

- Primarily focused on latency reduction
  - Bandwidth generally ignored
    - Prefetching can make it worse
  - Bandwidth is critical on modern machines (e.g., Origin 2000)

# Limitations of These Techniques

---

- Primarily focused on latency reduction
  - Bandwidth generally ignored
    - Prefetching can make it worse
  - Bandwidth is critical on modern machines (e.g., Origin 2000)
- Fail to deal with irregular codes
  - Static methods depend on predictable behavior
  - Prefetching can help some
    - Does not work as well on irregular as on regular
    - Does not strip mine to avoid multiple prefetches for same line

# Limitations of These Techniques

---

- Primarily focused on latency reduction
  - Bandwidth generally ignored
    - Prefetching can make it worse
  - Bandwidth is critical on modern machines (e.g., Origin 2000)
- Fail to deal with irregular codes
  - Static methods depend on predictable behavior
  - Prefetching can help some
    - Does not work as well on irregular as on regular
    - Does not strip mine to avoid multiple prefetches for same line
- New approach: focus on bandwidth
  - Continue to emphasize reuse
  - Don't throw bandwidth away in the memory hierarchy
    - Use as much of the cache line as possible

# Bandwidth as Limiting Factor

---

- Program and Machine Balance
  - **Program Balance:** Average number of bytes that must be transferred in memory per floating point operation
  - **Machine Balance:** Average number of bytes the machine can transfer from memory per floating point operation

# Bandwidth as Limiting Factor

- Program and Machine Balance
  - **Program Balance:** Average number of bytes that must be transferred in memory per floating point operation
  - **Machine Balance:** Average number of bytes the machine can transfer from memory per floating point operation

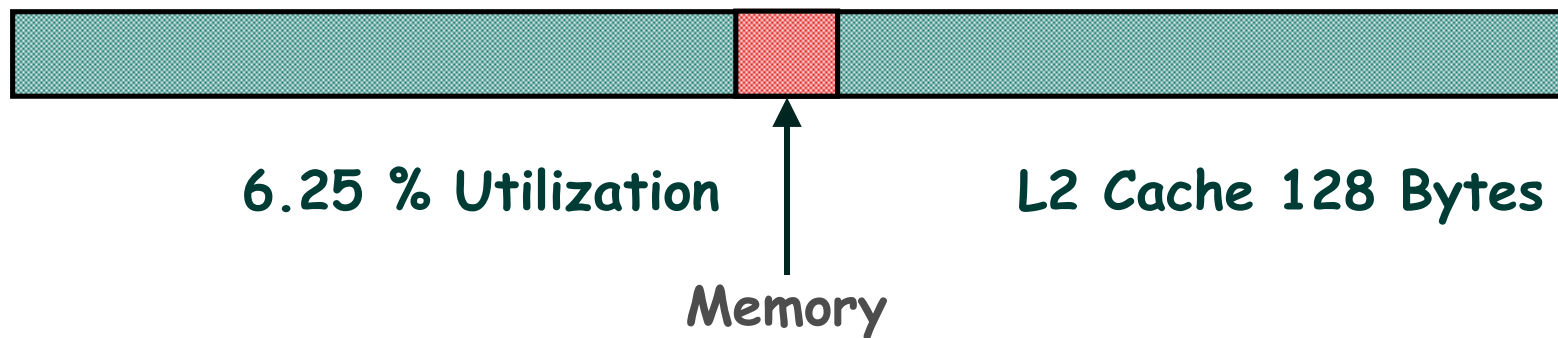
<i>Applications</i>	<i>Flops</i>	<i>L1-Reg</i>	<i>L2-L1</i>	<i>Mem-L2</i>
<i>Convolution</i>	1	6.4	5.1	5.2
<i>Dmxdpy</i>	1	8.3	8.3	8.4
<i>Mmiki (o2)</i>	1	24.0	8.2	5.9
<i>FFT</i>	1	8.3	3.0	2.7
<i>SP</i>	1	10.8	6.4	4.9
<i>Sweep3D</i>	1	15.0	9.1	7.8
<i>SGI Origin</i>	1	4	4	0.8

# Cache and Bandwidth

---

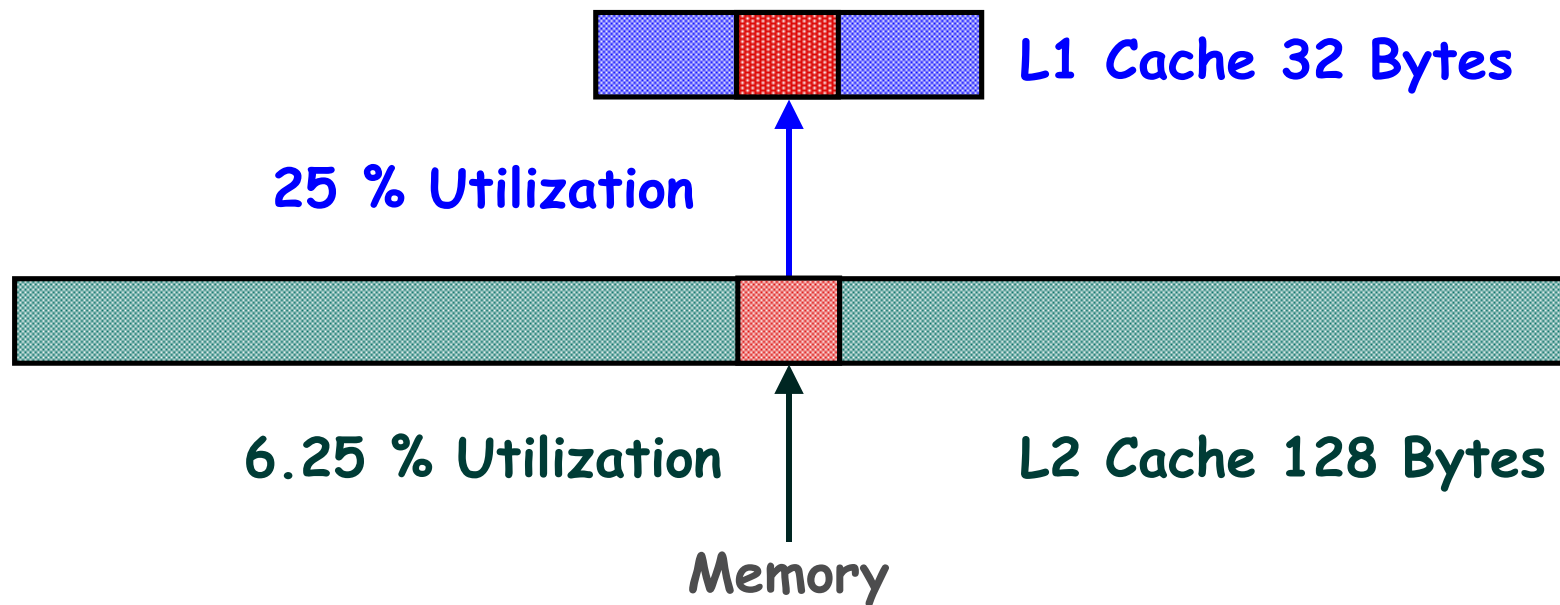
# Cache and Bandwidth

---



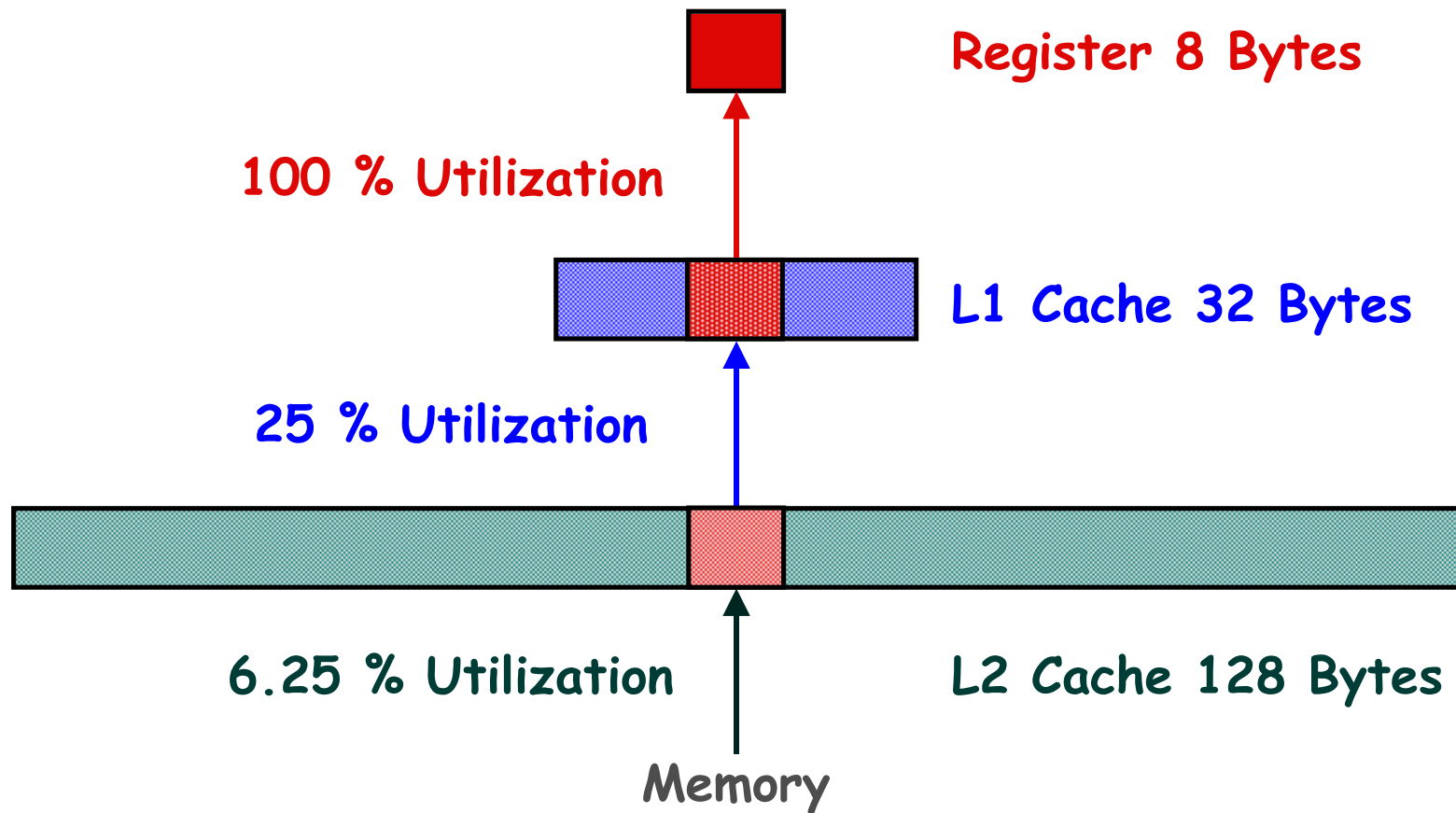
# Cache and Bandwidth

---



# Cache and Bandwidth

---



# Dynamic Data Packing

---

- Suppose the Calculation is Irregular
  - Example: Molecular Dynamics
    - Force calculations (pairs of forces)
    - Updating locations (single force per update)

# Dynamic Data Packing

---

- Suppose the Calculation is Irregular
  - Example: Molecular Dynamics
    - Force calculations (pairs of forces)
    - Updating locations (single force per update)
- Strategy
  - Dynamically reorganize data
    - So locations used together are updated together

# Dynamic Data Packing

---

- Suppose the Calculation is Irregular
  - Example: Molecular Dynamics
    - Force calculations (pairs of forces)
    - Updating locations (single force per update)
- Strategy
  - Dynamically reorganize data
    - So locations used together are updated together
  - Dynamically rewrite and reorganize interactions
    - So indirect accesses are not needed
    - So computation order has favorable performance characteristics

# Dynamic Data Packing

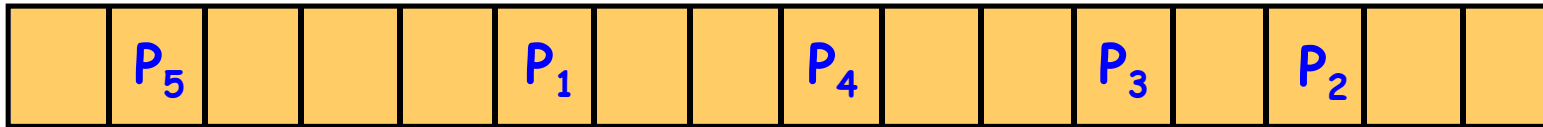
---

- **Suppose the Calculation is Irregular**
  - **Example: Molecular Dynamics**
    - Force calculations (pairs of forces)
    - Updating locations (single force per update)
- **Strategy**
  - **Dynamically reorganize data**
    - So locations used together are updated together
  - **Dynamically rewrite and reorganize interactions**
    - So indirect accesses are not needed
    - So computation order has favorable performance characteristics
  - **Example: “first touch”**
    - Assign elements to cache lines in order of first touch by pairs calculation

# First-Touch Ordering

---

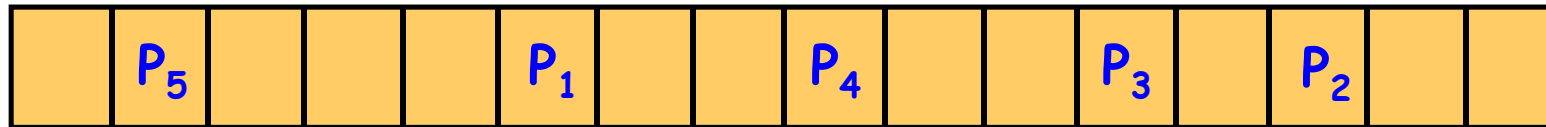
## Original Ordering



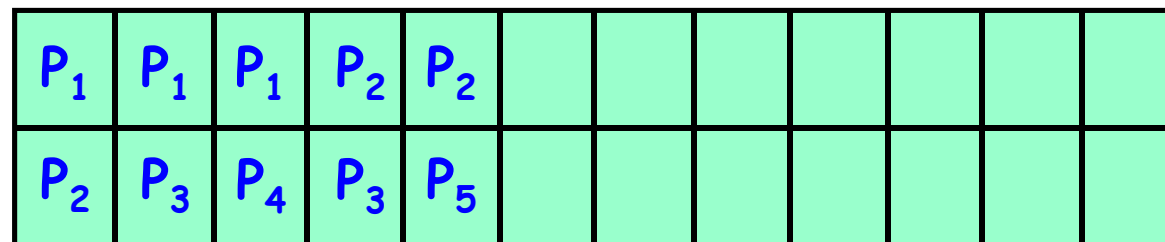
# First-Touch Ordering

---

## Original Ordering



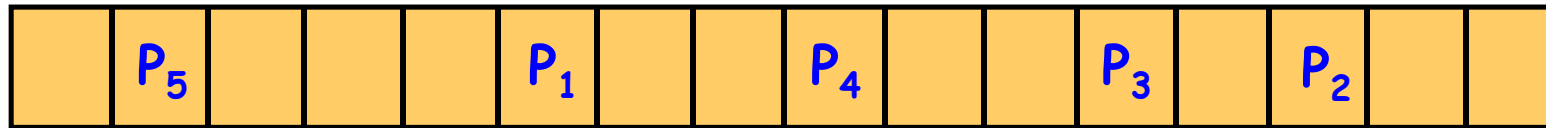
## Interaction Pairs



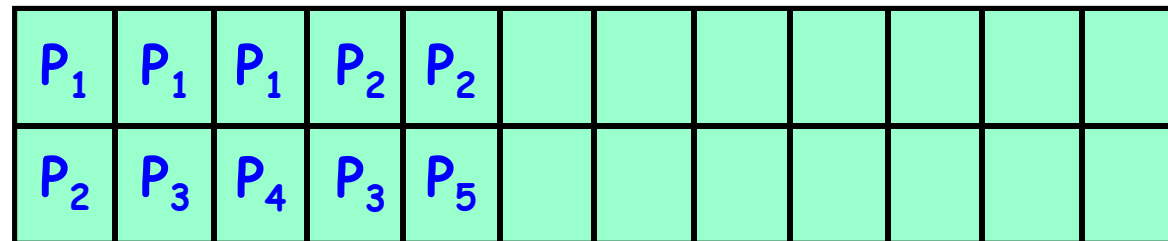
# First-Touch Ordering

---

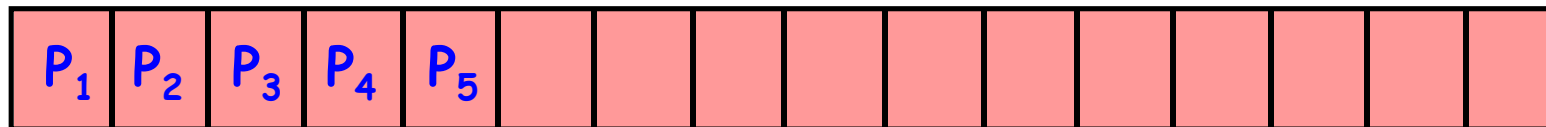
## Original Ordering



## Interaction Pairs



## First-Touch Ordering



# First Touch Data Reordering

---

```
DO I = 1, Npairs
    F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
    F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L1(I)))
ENDDO
DO I = 1, Nparticles
    A(I) = g(A(I), F(I))
ENDDO
```

# First Touch Data Reordering

---

```
DO I = 1, Npairs
    F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
    F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L1(I)))
ENDDO
DO I = 1, Nparticles
    A(I) = g(A(I), F(I))
ENDDO
```

After data reordering:

```
DO I = 1, Npairs
    F(L(L1(I))) = F(L(L1(I))) + f(A(L(L1(I))), A(L(L2(I))))
    F(L(L2(I))) = F(L(L2(I))) + f(A(L(L2(I))), A(L(L2(I))))
ENDDO
DO I = 1, Nparticles
    A(L(I)) = g(A(L(I)), F(L(I)))
ENDDO
```


# First Touch Data Reordering

---

```
DO I = 1, Npairs
    F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
    F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L1(I)))
ENDDO
DO I = 1, Nparticles
    A(I) = g(A(I), F(I))
ENDDO
```

After data reordering:

```
DO I = 1, Npairs
    F(L(L1(I))) = F(L(L1(I))) + f(A(L(L1(I))), A(L(L2(I))))
    F(L(L2(I))) = F(L(L2(I))) + f(A(L(L2(I))), A(L(L2(I))))
ENDDO
DO I = 1, Nparticles
    A(L(I)) = g(A(L(I)), F(L(I)))
ENDDO
```



Extra level of indirection!

# First Touch Data Reordering

---

```
DO I = 1, Npairs
    F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
    F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L1(I)))
ENDDO
DO I = 1, Nparticles
    A(I) = g(A(I), F(I))
ENDDO
```

After data reordering:

```
DO I = 1, Npairs
    F(L(L1(I))) = F(L(L1(I))) + f(A(L(L1(I))), A(L(L2(I))))
    F(L(L2(I))) = F(L(L2(I))) + f(A(L(L2(I))), A(L(L2(I))))
ENDDO
DO I = 1, Nparticles
    A(L(I)) = g(A(L(I)), F(L(I)))
ENDDO
```

↑  
Can be composed!

# First Touch Data Reordering

---

## Redefine L1 and L2

```
DO I = 1, Npairs
  L1(I) = L(L1(I))
  L2(I) = L(L2(I))
```

Done once for all steps!

```
ENDDO
```

```
DO I = 1, Npairs
  F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
  F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L2(I)))
```

```
ENDDO
```

```
DO I = 1, Nparticles
  A(L(I)) = g(A(L(I)), F(L(I)))
```

```
ENDDO
```

# First Touch Data Reordering

---

## Redefine L1 and L2

And reorder location updates...

```
DO I = 1, Npairs
  L1(I) = L(L1(I))
  L2(I) = L(L2(I))
```

Done once for all steps!

```
ENDDO
```

```
DO I = 1, Npairs
  F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
  F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L2(I)))
```

```
ENDDO
```

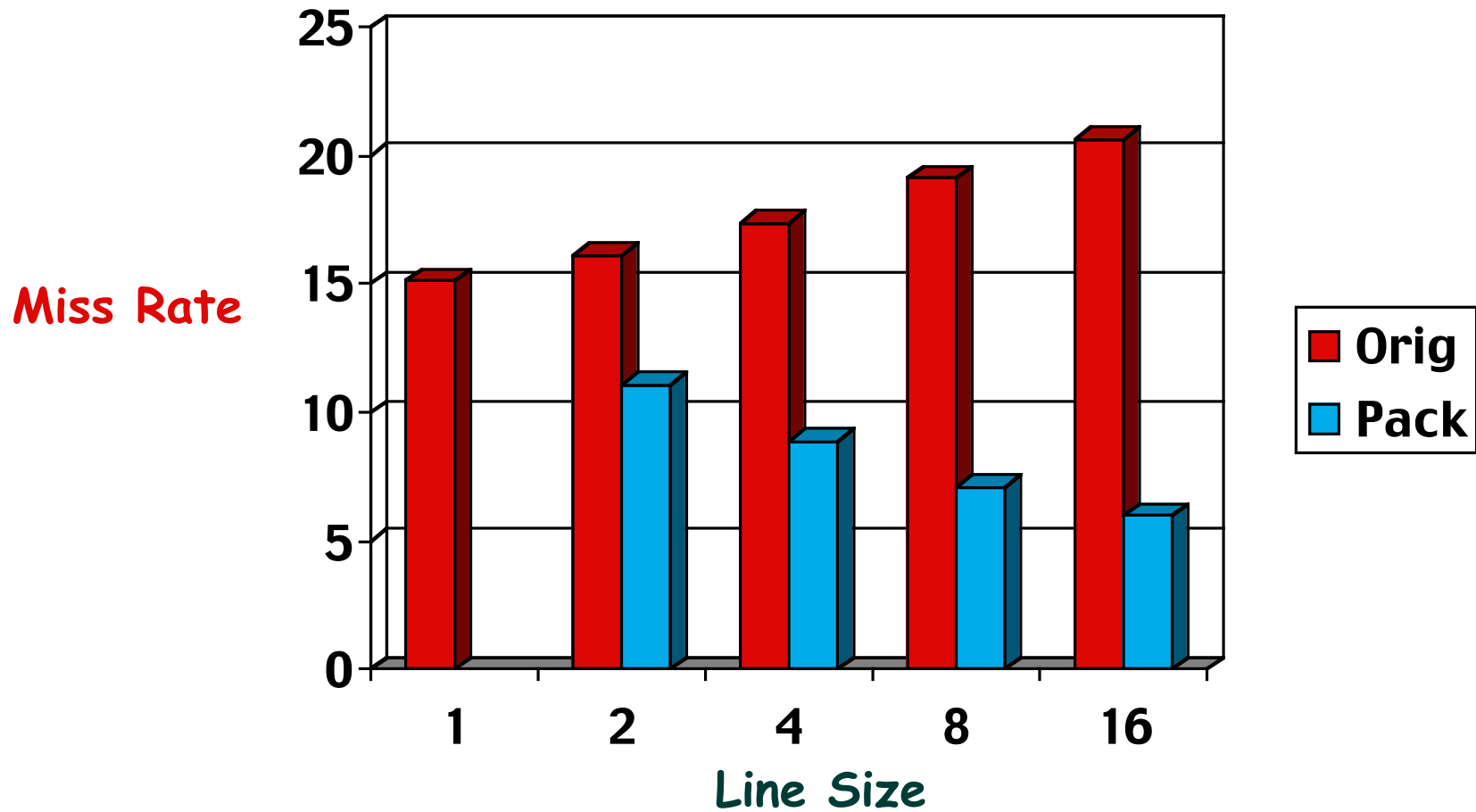
```
DO I = 1, Nparticles
  A(I) = g(A(I), F(I))
```

Independent of update order

```
ENDDO
```

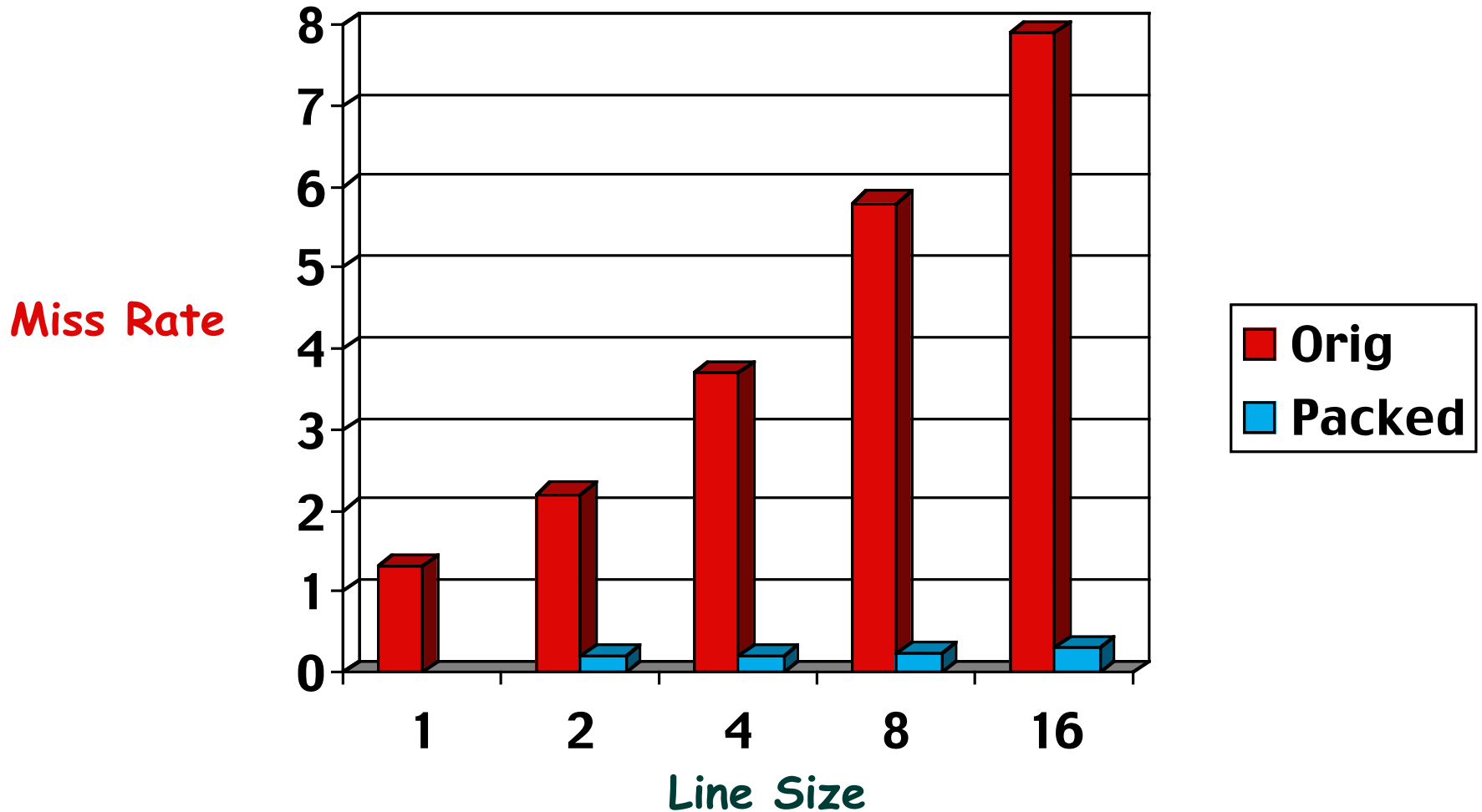
# Results of Packing 1

8K Molecules, 2K Cache



# Results of Packing 2

8K Molecules, 4K Cache



# Irregular Blocking

---

Suppose we employ a different ordering for data, e.g., Hilbert

```
DO I = 1, Npairs
  F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
  F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L1(I)))
ENDDO
```

# Irregular Blocking

---

Suppose we employ a different ordering for data, e.g., Hilbert

```
DO I = 1, Npairs
  F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
  F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L1(I)))
ENDDO
```

Computation can be ordered by sorting the pairs by Hilbert index:

# Hilbert Ordering

---

Suppose we employ a different ordering for data, e.g., Hilbert

```
DO I = 1, Npairs
  F(L1(I)) = F(L1(I)) + f(A(L1(I)), A(L2(I)))
  F(L2(I)) = F(L2(I)) + f(A(L2(I)), A(L1(I)))
ENDDO
```

Computation can be ordered by sorting the pairs by Hilbert index:

```
DO p1 = 1, Nparticles
  FOR p2 in interacts_with(p1)
    F(p1) = F(p1) + f(A(p1), A(p2))
    F(p2) = F(p2) + f(A(p2), A(p1))
  ENDFOR
ENDDO
```

**Abstract  
view**

# Hilbert Ordering II

---

```
DO p1 = 1, Nparticles
  FOR p2 in interacts_with(p1)
    F(p1) = F(p1) + f(A(p1), A(p2))
    F(p2) = F(p2) + f(A(p2), A(p1))
  ENDFOR
ENDDO
```

Problem: TLB misses

# Hilbert Ordering II

---

```
DO p1 = 1, Nparticles
  FOR p2 in interacts_with(p1)
    F(p1) = F(p1) + f(A(p1), A(p2))
    F(p2) = F(p2) + f(A(p2), A(p1))
  ENDFOR
ENDDO
```

**Problem: TLB misses**

**Solution: Blocking**

# Hilbert Ordering II

---

```
DO p1 = 1, Nparticles
  FOR p2 in interacts_with(p1)
    F(p1) = F(p1) + f(A(p1), A(p2))
    F(p2) = F(p2) + f(A(p2), A(p1))
  ENDFOR
ENDDO
```

Blocked  
1 Level

```
DO b1 = 1, Nblocks
  FOR b2 in block_interacts_with(b1)
    FOR p1 in b1
      FOR p2 in (b2 ∩ interacts_with(p1))
        F(p1) = F(p1) + f(A(p1), A(p2))
        F(p2) = F(p2) + f(A(p2), A(p1))
      ENDFOR
    ENDFOR
  ENDFOR
ENDDO
```

# Multilevel Blocking Algorithm

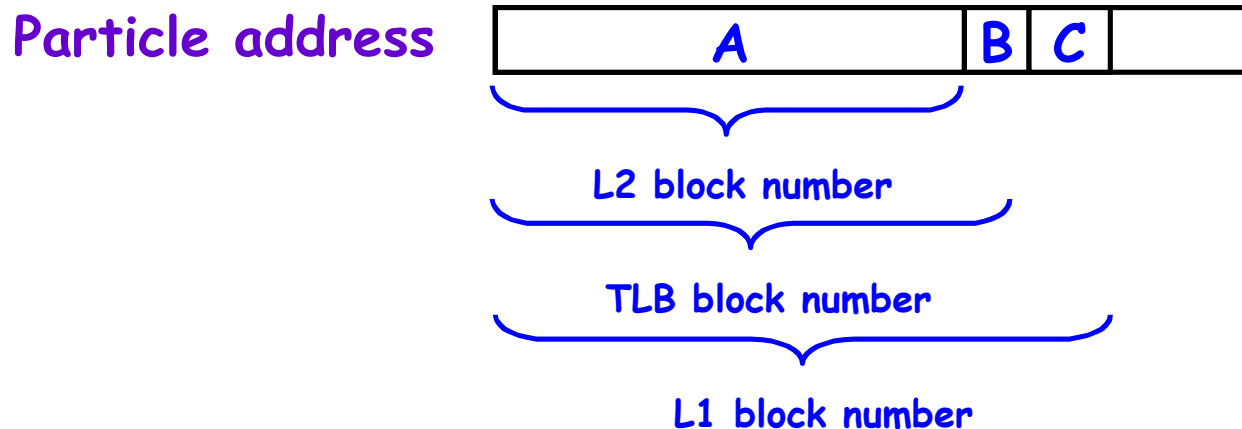
---

- Reorder data according to the chosen scheme
  - One promising strategy:
    - Hilbert order (space-filling curve)

# Multilevel Blocking Algorithm

---

- Reorder data according to the chosen scheme
  - One promising strategy:
    - Hilbert order (space-filling curve)
- Associate a tuple of block numbers with each particle
  - One integer number per level of the memory hierarchy
    - Block number = selected bits of particle address



# Multilevel Blocking Algorithm

---

- Reorder data according to the chosen scheme
  - One promising strategy:
    - Hilbert order (space-filling curve)
- Associate a tuple of block numbers with each particle
  - One integer number per level of the memory hierarchy
    - Block number = selected bits of particle address
- Associate a block tuple with each element of the interaction list
  - Interleave the block numbers of the pairs



# Multilevel Blocking Algorithm

---

- Reorder data according to the chosen scheme
  - One promising strategy:
    - Hilbert order (space-filling curve)
- Associate a tuple of block numbers with each particle
  - One integer number per level of the memory hierarchy
    - Block number = selected bits of particle address
- Associate a block tuple with each element of the interaction list
  - Interleave the block numbers of the pairs
- Sort the interaction list lexicographically by tuples (linear time!)
  - This automatically carries out the blocking at multiple levels

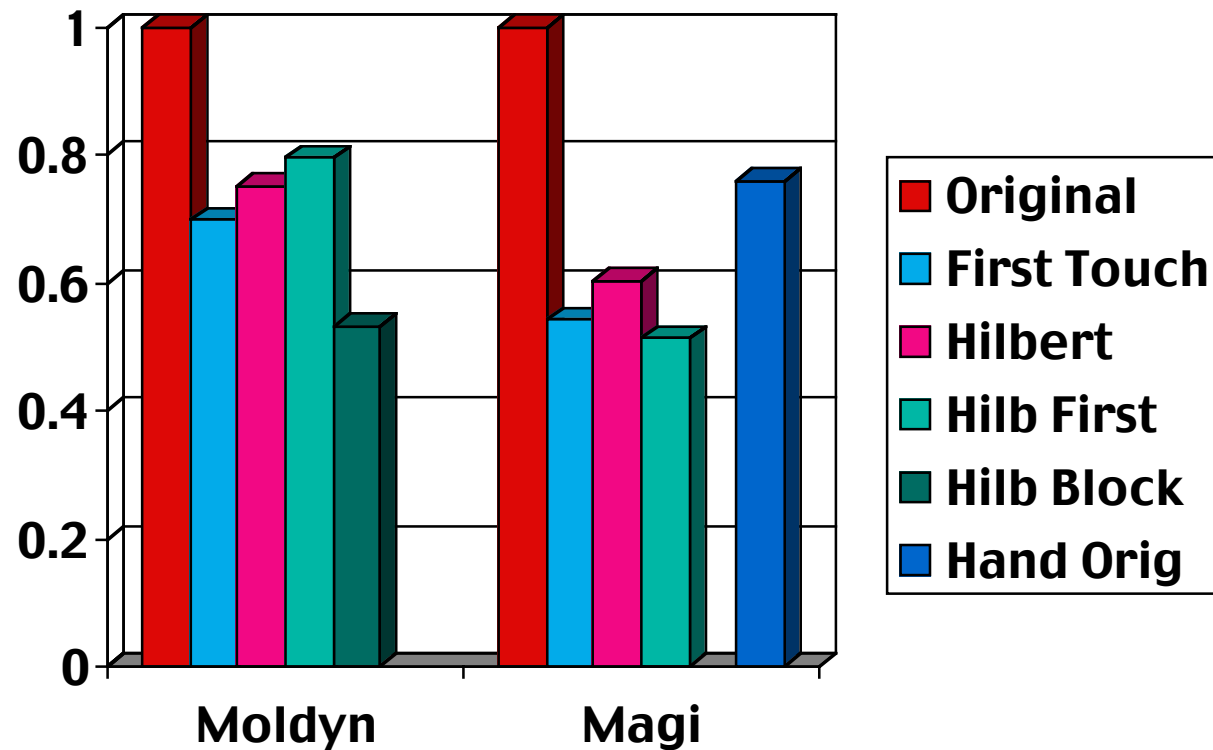
# Multilevel Blocking Algorithm

---

- Reorder data according to the chosen scheme
  - One promising strategy:
    - Hilbert order (space-filling curve)
- Associate a tuple of block numbers with each particle
  - One integer number per level of the memory hierarchy
    - Block number = selected bits of particle address
- Associate a block tuple with each element of the interaction list
  - Interleave the block numbers of the pairs
- Sort the interaction list lexicographically by tuples (linear time!)
  - This automatically carries out the blocking at multiple levels
- Perform the calculation by traversing the interaction list

# Data and Computation Reordering

- Results:
  - Run on unloaded SGI with hardware counters



# Summary

---

- PITAC has made the case for funding
  - High end computing is special concern of government
    - Research will be driven by petaflops/petaops goal
  - Software is a critical component

# Summary

---

- PITAC has made the case for funding
  - High end computing is special concern of government
    - Research will be driven by petaflops/petaops goal
  - Software is a critical component
- Petaflops computers present deep challenges for software
  - Unprecedented levels of parallelism
  - Memory hierarchies much deeper
  - Components may be heterogeneous
  - Applications will be more complex
  - Tools will be essential

# Summary

---

- PITAC has made the case for funding
  - High end computing is special concern of government
    - Research will be driven by petaflops/petaops goal
  - Software is a critical component
- Petaflops computers present deep challenges for software
  - Unprecedented levels of parallelism
  - Memory hierarchies much deeper
  - Components may be heterogeneous
  - Applications will be more complex
  - Tools will be essential
- Much more work needed in memory hierarchies
  - Bandwidth emerging as a critical issue
  - Strategies for regular problems must be adapted to irregular