# SOFTWARE CRISIS 2000+

# Software failures 2015/2016

- **Nest thermostat leaves users in the cold**
- **600,000 RBS payments go missing**
- **US Airforce F-15**
  - **Problems with software left pilot radar blind**
  - **400 Billion USD**
  - **8 million lines**
  - **“the rate of deficiency correction has not kept pace with the discovery rate” (DoD)**

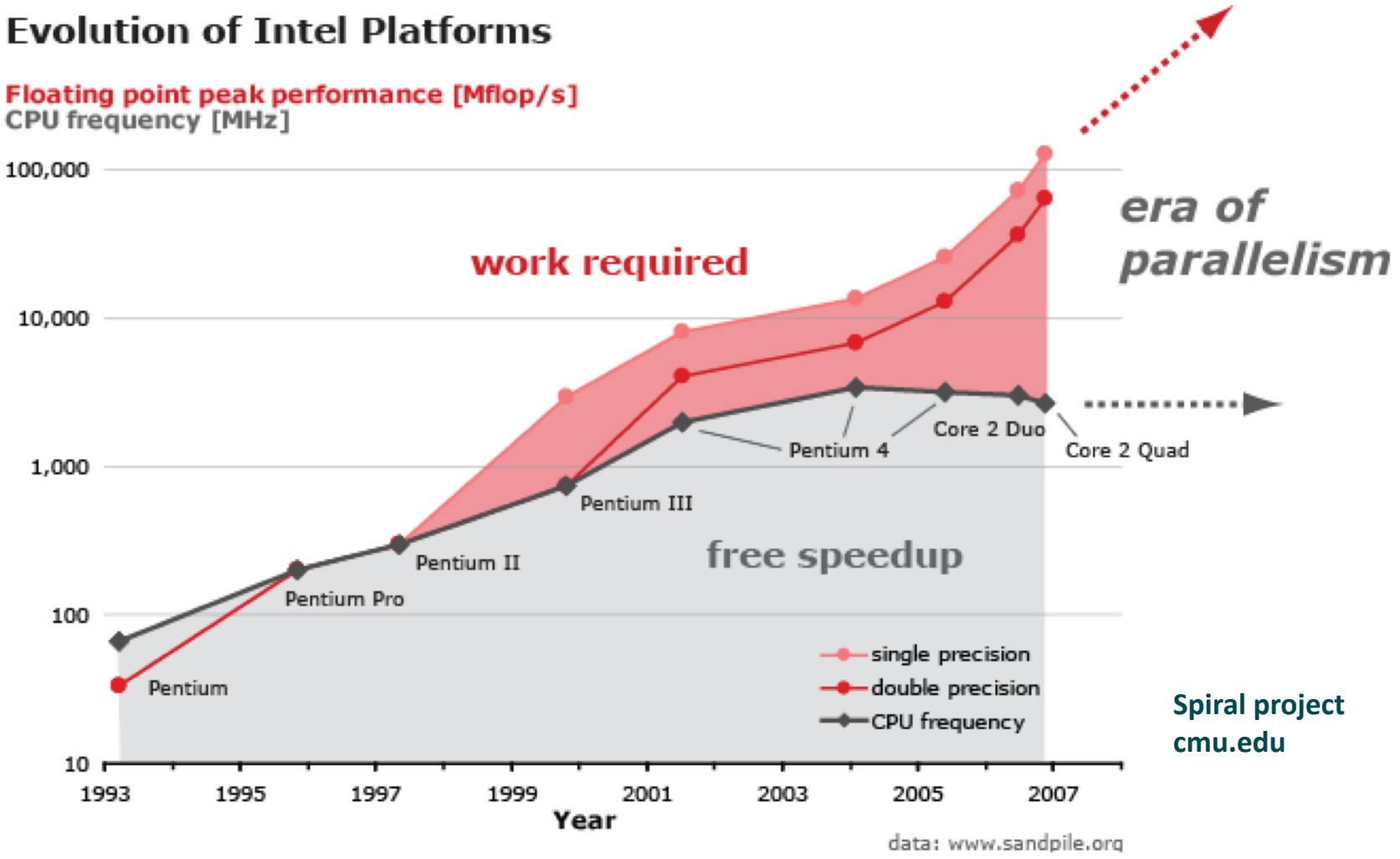
# From year 2000

- $2 million dollars bought what can be got now for $10,000
- Computers are everywhere and connected
- So given this power why can computers not solve any problem?
  - Many problems not clearly understood computationally
  - Many problems scale badly
  - New computer hardware/config needs new software methods (e.g. neural-net, quantum computers, grid computing) .. non Von-Neumann architecture

# New domains

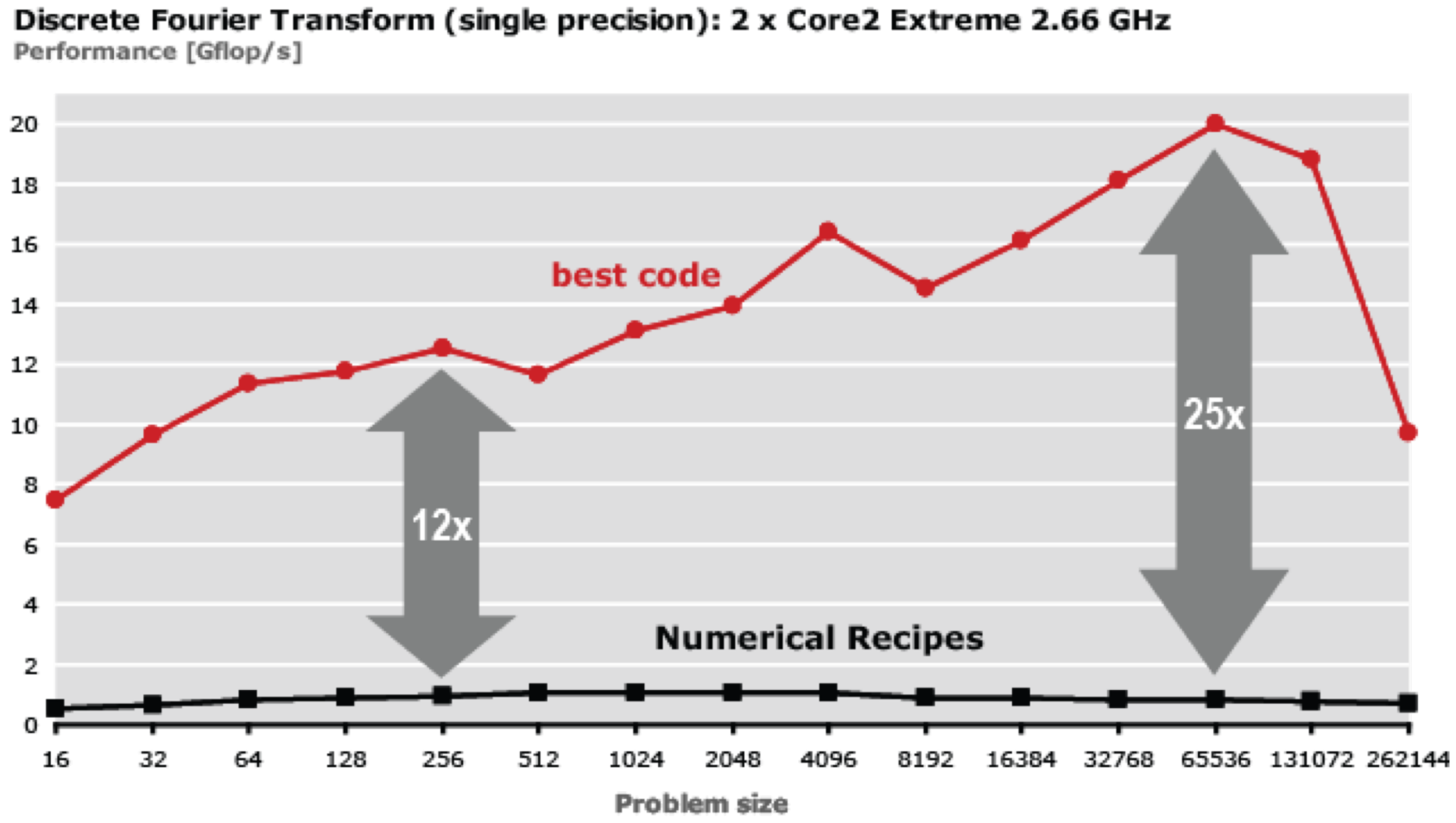
- Architecture
  - Building design packages
- Medical
  - Expert diagnostic help (AI)
  - Simulation (heart disease)
- Law
  - Contract analysis and case law database (AI, clustering)
- Electronics
  - Simulation
- DNA analysis
  - Clustering and big data analysis

# Moore's law and parallelism




**Spiral project**
**cmu.edu**

# Performance Challenges of modern software



# Parallelism simple example

```
int addElements(int data[]) {
   for (int i=0;i<array.length;i++) {
      sum=sum+data[i];
   }
}
```

However many processors are used, its not obvious how this code's speed will be increased.

# Parallism simple example (threaded)

```
class Adder extends Thread {
  int sum=0;
  int index1=0,index2=0;
  public Adder(int data[], int index1,int index2) {
     this.data=data;
     this.index1=index1;
     this.index2=index2;
  }
  int addElements() {
    sum=0;
    for (int i=index1;i<index2;i++) {
      sum=sum+data[i];
    }
    return(sum);
  }
  int getSum() {
     return(sum);
  }
  public void run() {
     addElements();
  }
}
```

# Parallism simple example (threaded)

```
int addElements(int data[]) {
   Adder adder1=new Adder(data,0,data.length/2);
   Adder adder2=new Adder(data,data.length/2+1,data.length);
   adder1.start();
   adder2.start();
   adder1.join();
   adder2.join();
   return(adder1.getSum()+adder2.getSum());
}
```

# Spiral Project www.spiral.net

- Computers used to generate code for multi-core architecture, which then performs functions such as Fast Fourier Transform
- www.spiral.net/codegenerator.html
- Spiral also produces software to automatically designs hardware cores
- By automating hardware/software implementations accidental complexity can be factored out of design

# What is different now?

- Software is developed to be sold shrink wrapped
- Users develop niche products not systems
- Simple business applications have vanished to be done with standard packages

# What remains the same

- Centrality of the gifted programmer concentrating on the conceptual integrity of the system
- The difficult bit is still the conceptual design
- The constraints (the box defined by resources)

# Software Engineer roles

- Used to be broken up
  - System Analyst (sales)
  - Database designer
  - Programmer
  - Tester
- Now commonly all 1 role
  - SQL, web software developer
  - Tester and analyst

# Software Engineering 2016+

- Most software now also on mobile devices
- Fragmentation of platforms (need for portability)
  - iPhone
  - Windows
  - Android
  - Blackberry
- Connectivity now broad and narrowband and wireless

# Mobile development

- Fragmentation
  - Want 1 code base, many platforms
  - Example PhoneGap/Cordova
    - Development in Javascript
    - APIs connect mobile API to Javascript
- Diversity of mobile web development
  - Issues with connectivity
  - Battery life should ideally be conservered

# Current challenges

- Ubiquitous computing
- Intelligent sensing: voice, vision, touch
- Massive connectivity
- Super scalability
- Critical systems
- Secure systems
- Configuration modelling

# Ubiquitous computing and intelligent sensing

- Computing implemented everywhere
- Technologies such as
  - Plastic transistor technology
  - Wearable computers
- Intelligent sensing
  - Ability of systems to see/hear understand their environment
  - Haptic feedback, advanced interfaces
  - Sense substitution technologies. e.g. seeing through your tongue

# Scaling up

- Scale up
  - Increase process/memory speed etc.
- Benefits
  - Simple software architecture
- Problems
  - Cost per increase
  - Limitations of scalability
  - Hardware failure leads to total loss of node

# Scaling

- Scale out

  - Benefits
    - Cheap, easier to have failsafe, can be geographically separated
  - Issues
    - Software engineering has to be more complex

# Massive scalability and connectivity

- Allows for the implementation of grid computing
  - Loosely coupled, heterogeneous, geographically dispersed
  - Issues of trust, (node to cluster level and vice-versa)
  - Differences of OS, language, capacity
  - Rationing of capacity between grid task and local tasks
  - Loss of network or loss of node (power off)
- Super scalability
  - Cloud computing
  - Scaling transparency
- Modelling co-current systems
  - Actor model

# Critical and secure systems

- Safety critical
  - Highly structured design/development methodology: limiting function/method length, restricting scope, limit on pointer dereferencing
  - Fail safe system design (limit checking driver code.. think of Ariane), clear modular decomposition/ responsibilities
  - Self healing systems, fault tolerant systems
- Secure systems
  - Utilizing latest encryption and signature technolgies
  - Hardware/software advances such as biometrics
  - Again many advances and experiences, see http://www.sans.org/top25-software-errors

# Configuration modelling

- System self configures, example due to
  - Demand
  - Fault occurrence
  - Costs
- Common in networks
  - IP routing
- One example of complex demand management is power grids
  - Power cables warm up when drawing a lot of current, causing sag
  - If the line drops too much it is shut down, causes load to increase on other lines, sometimes leading to blackouts
  - Software to run smart power grids is needed to improve performance and reliability
  - With the increase in distributed power generation e.g. wind power this has become more of an issue

# Summary

- New domains
- More complex software
- Massive degree of independency
- More parallel architectures
- Greater security threats