I. Introduction

Content

- Parallel computers
- Why parallel computing
- Application examples
- Short history
- To port or not to port.
- Performance of parallel systems

Parallel computers

Parallel computer

- A *parallel computer* is a collection of processing elements that cooperate and communicate to solve large problems fast

=>Questions:

1. How large a collection are we talking about?
2. How powerful are the individual processing elements and can the number be increased in a straight-forward manner?
3. How do they cooperate and communicate?
4. How are data transmitted between processors, what sort of interconnection is provided, and what operations are available to sequence the actions carried out on different processors?
5. What are the primitive abstractions that the hardware and software provide to the programmer?
6. How does it all translate into performance?

General comments

- parallel machines occupy a rich and diverse design space: from small to very large
- field of parallel processing:
  - concerned with architectural and algorithmic methods for enhancing the performance or other attributes (e.g. reliability) of computers through various forms of concurrency
- History changes:
  - 30 years ago: parallel computing has emerged from the enclaves of research institutions and cutting edge technology firms
    - mid1990s: rare and for solving critical problems
  - today most personal computers arrive prebuilt with multiple processing units:
    - proliferation of 64 processor systems will emerge soon
    - multiple processor cores on a single silicon die, parallel computers are becoming ubiquitous.

Term clarification

- Parallel processing, in the *literal sense* of the term, is used in virtually every modern computer:
  - overlapping I/O with computation is a form of parallel processing,
  - the overlap between instruction preparation and execution in a pipelined processor.
  - use of multiple functional units (e.g., separate integer and floating-point ALUs or two floating-point multipliers in one ALU)
  - multitasking (which allows overlap between computation and memory load necessitated by a page fault).
  - Horizontal microprogramming, and its higher-level incarnation in very-long-instruction-word (VLIW) computers, also allows some parallelism.
- In this lecture,

  the term *parallel processing* is used in a restricted sense:
  - of having multiple (usually identical) processors for the main computation and not for the I/O or other peripheral activities.

Why parallel computing?

Big effort

Development of parallel software has traditionally been thought of as time and effort intensive, due to:

- inherent complexity of specifying and coordinating concurrent tasks,
- a lack of portable algorithms, standardized environments, and software development toolkits.

Motivations for parallel processing

1. **Higher speed, or solving problems faster**.
   - This is important when applications have “hard” or “soft” deadlines.
   - For example, we have at most a few hours of computation time to do 24-hour weather forecasting or to produce timely tornado warnings.
2. **Higher computational power, or solving larger problems.**
   - Carry out simulation runs for longer periods of time (e.g., 5-day, as opposed to 24-hour, weather forecasting).
3. **Higher throughput, or solving more instances of given problems.**
   - Important when many similar tasks must be performed.
   - For example, banks and airlines, among others, use transaction processing systems that handle large volumes of data.

Ultimate figure-of-merit: speed-up

- The computation ***speed-up*** factor with respect to a uniprocessor.
  - The ultimate efficiency in parallel systems is to achieve a computation speed-up factor of *p* with *p* processors.
  - Although in many cases this ideal cannot be achieved, *some* speed-up is generally possible.
  - The actual gain in speed depends on the architecture used for the system and the algorithm run on it.
- For a task that is impossible to perform on a single processor in view of its excessive running time, the computation speed-up factor can rightly be taken to be larger than *p* or even infinite.
  - Analogue of several men moving a heavy piece of machinery or furniture in a few minutes, whereas one of them could not move it at all,
  - Referred as ***parallel synergy***.

Faster = better?

- Market advantage
- Save lives
- Make the impossible possible
  - Environmental modeling
  - Space exploration
  - Biological reasearch

Moore's law

- The growth of microprocessor speed/performance by a factor of 2 every 18 months (or about 60% per year) is known as **Moore's law.**
- This growth is the result of a combination of two factors:

1. Increase in complexity (related both to higher device density and to larger size) of VLSI chips
2. Introduction of, and improvements in, architectural features such as
   - on-chip cache memories, large instruction buffers, multiple instruction issue per cycle, multithreading, deep pipelines, out-of-order instruction execution.

Moore's law: based on empirics

- 1965: reasoning was based on an empirical log-linear relationship between device complexity and time, observed over three data points.
- the empirical relationship has been amazingly resilient over the years both for microprocessors as well as for DRAMs.
- The law has been extrapolated to state that the amount of computing power available at a given cost doubles approximately every 18 months.

Limitations of Moore's law

- physical limit is imposed by the finite speed of signal propagation along a wire:
  - This is sometimes referred to as the **speed-of-light argument (or limit)**
- In the context of parallel computing motivation, this is **the computational power argument**

Speed-of-light argument

- The speed of light is about 30 cm/ns.
- Signals travel on a wire at a fraction of the speed of light.
- If the chip diameter is 3 cm, say, any computation that involves signal transmission from one end of the chip to another cannot be executed faster than 10^{10} times per second.
- Reducing distances by a factor of 10 or even 100 will only increase the limit by these factors; we still cannot go beyond 10^{12} computations per second.
- To relate the above limit to the instruction execution rate (MIPS or FLOPS), we need to estimate the distance that signals *must* travel within an instruction cycle.
- We are in fact not very far from limits imposed by the speed of signal propagation and several other physical laws.

Consequence

- The speed-of-light argument suggests that once the above limit has been reached, the **only path** to improved performance is the use of multiple processors.
- Of course, the same argument can be invoked to conclude that any parallel processor will also be **limited by the speed at which the various processors can communicate** with each other.
  - However, because such communication does not have to occur for every low-level computation, the limit is less serious here.
  - For many applications, a large number of computation steps can be performed between two successive communication steps, thus amortizing the communication overhead.

The Memory/Disk Speed Argument

- The overall speed of computation is determined not just by the speed of the processor, but also by the ability of the memory system to feed data to it.
- While clock rates of high-end processors have increased at roughly 40% per year over the past decade, DRAM access times have only improved at the rate of roughly 10% per year over this interval.
- Coupled with increases in instructions executed per clock cycle, this gap between processor speed and memory presents a tremendous performance bottleneck.

The Memory/Disk Speed Argument

- The overall performance of the memory system is determined by the fraction of the total memory requests that can be satisfied from the cache.
- Parallel platforms typically yield better memory system performance because they provide

(i) larger aggregate caches, and

(ii) higher aggregate bandwidth to the memory system (both typically linear in the number of processors).

- The heart of parallel algorithms: locality of data reference

Data Communication Argument

- As the networking infrastructure evolves, the vision of using the Internet as one large heterogeneous parallel/distributed computing environment has begun to take shape.
- In many applications there are constraints on the location of data and/or resources across the Internet.
  - An example of such an application is mining of large commercial datasets distributed over a relatively low bandwidth network.
  - In such applications, even if the computing power is available to accomplish the required task without resorting to parallel computing, it is infeasible to collect the data at a central location.
  - In these cases, the motivation for parallelism comes not just from the need for computing resources but also from the infeasibility or undesirability of alternate (centralized) approaches.

Application examples

There exists a need to solve large problems

- Earth environment prediction
- Nuclear weapons testing
- Quantum chemistry
- Computational biology
- Data mining for large and very large data sets
- Astronomy and cosmology
- Cryptography
- Approximate algorithms for *NP*-complete problems

etc

Computer animation

- rendering is the step where information from the animation files, such as lighting, textures, and shading, is applied to 3D models to generate the 2D image that makes up a frame of the film.
- parallel computing is essential to generate the needed no. of frames (24 fps) for a feature length film.
- 1995 - Pixar: Toy Story, the first completely computer generated feature length film, was processed on a "renderfarm" consisting of 100 dual processor machines.
- 1999 – Pixar: Toy Story 2, a 1400 processor system with the improvement in processing power fully reflected in the improved details in textures, clothing, and atmospheric effects.
- 2001: Monsters, Inc. used a system of 250 enterprise servers each containing 14 processors for a total of 3500 processors.

Biological sciences

- have taken dramatic leaps forward with the availability of DNA sequence information from a variety of organisms, including humans.
- Celera Corp.: whole genome shotgun algorithm.
  - The idea is to break the genome into small segments, experimentally determine the DNA sequences of the segments, and then use a computer to construct the entire sequence from the segments by finding overlapping areas.
  - The computing facilities used by Celera to sequence the human genome included 150 four way servers plus a server with 16 processors and 64GB of memory.
  - The calculation involved 500 million trillion base to base comparisons.

Astrophysics

- explored the evolution of galaxies, thermonuclear processes, and the analysis of extremely large datasets from telescopes.
- analyzing extremely large datasets.
  - Sky Survey datasets (such as the Sloan Digital Sky Surveys) represent some of the largest scientific datasets.
  - Effectively analyzing these datasets requires tremendous computational power

Numerical simulations – an example

- To learn how the southern oceans transport heat to the South Pole, the following model has been developed:
  - The ocean is divided into 4096 regions E–W, 1024 regions N–S, and 12 layers in depth (50 M 3D cells).
  - A single iteration of the model simulates ocean circulation for 10 minutes and involves about 30B floating-point operations.
  - To carry out the simulation for 1 year, about 50,000 iterations are required.
  - Simulation for 6 years would involve 10^{16} floating-point operations.

Engineering and Design

- design of airfoils: optimizing lift, drag, stability,
- internal combustion engines: optimizing charge distribution, burn,
- high-speed circuits: layouts for delays and capacitive and inductive effects,
- structures: optimizing structural integrity, design parameters, cost,
- design of microelectromechanical and nanoelectromechanical systems

Commercial applications

- cost-effective servers capable of providing scalable performance.
  - Parallel platforms ranging from multiprocessors to Linux clusters are frequently used as web and database servers.
  - Large brokerage houses on Wall Street handle hundreds of thousands of simultaneous user sessions and millions of orders.
  - While not highly visible, some of the largest supercomputing networks are housed on Wall Street.
  - Platforms such as IBMs SP supercomputers and Sun Ultra HPC servers power these business-critical sites.
- The availability of large-scale transaction data has also sparked considerable interest in data mining and analysis for optimizing business and marketing decisions:
  - Geographically distributed nature of this data require the use of parallel algorithms for such problems as association rule mining, clustering, classification, and time-series analysis.

Computer Systems

- In the case of network intrusion detection, data is collected at distributed sites and must be analyzed rapidly for signaling intrusion.
- The infeasibility of collecting this data at a central location for analysis requires effective parallel and distributed algorithms.
- In the area of cryptography, some of the most spectacular applications of Internet-based parallel computing have focused on factoring extremely large integers.
- A modern automobile consists of tens of processors communicating to perform complex tasks for optimizing handling and performance.

Short history

Ups and downs | The beginning

- The history of parallel processing has had its ups and downs (company formations and bankruptcies) with what appears to be a 20-year cycle.
- Serious interest in parallel processing started in 1960s:
  - ILLIAC IV,
    - designed at the University of Illinois
    - later built and operated by Burroughs Corporation,
    - the first large-scale parallel computer implemented;
    - its 2D-mesh architecture with a common control unit for all processors was based ontheories developed in the late 1950s.
    - It was to scale to 256 processors (4 quadrants of 64 procs each).
    - Only one 64-processor quadrant was eventually built,
    - It clearly demonstrated the feasibility of highly parallel computers
    - revealed some of the difficulties in their use.

Commercial interest

- resurfaced in the 1980s:
  - Driven primarily by contracts from the defense establishment and other federal agencies in USA, numerous companies were formed to develop parallel systems.
- Three factors led to another recess:

  1. Government funding in the United States and other countries dried up, in part related to the end of the cold war.
  2. Commercial users in banking and other data-intensive industries were either saturated or disappointed by application difficulties.
  3. Microprocessors developed so fast in terms of performance/cost ratio that custom designed parallel machines always lagged in cost-effectiveness.

  => Many of the newly formed companies went bankrupt or shifted their focus to developing software for distributed (workstation cluster) applications.

Turning points

- Latest 60s: remarkable turnover of vendors, architectures, technologies, and systems usage.
- Second half of 70s: introduction of vector computer systems marked the beginning of modern supercomputing
- First half of the 1980s the integration of vector systems into conventional computing environments
- Late 1980s: “attack of the killer micros” => usage of “off-the-shelf” microprocessors instead of custom processors for massively parallel systems
- Early 1990s, a new generation of massively parallel processor (MPP) systems came on the market, claiming to equal or even surpass the performance of vector multiprocessors.
- June 1993: Top500 list was begun to provide a more reliable basis for statistics on high-performance computers
- 1994: SGI, Digital & Sun began selling symmetric multiprocessor (SMP) models in workstation families for industrial customers (e.g. IBM SP2)

Now we are in an up period!

- Driven by the Internet revolution and its associated “information providers,” a 3rd resurgence of parallel architectures is currently present.
- Centralized, high-performance machines may be needed to satisfy the information processing/access needs of some of these providers.
- Parallel computing is more than just a strategy for achieving high performance— it is a compelling vision for how computation can seamlessly scale from a single processor to virtually limitless computing power.

To port or not to port

Porting a code to parallel architectures

- is more than simply bringing up an existing code on a new machine.
  - parallel machines are fundamentally different from their vector predecessors,
  - porting presents an opportunity to
  1. reformulate the basic code and data structures
  2. reassess the basic representation of the processes or dynamics involved.

Difficulty of the parallel programming task

- the expression of an explicitly parallel program is difficult
  - The developer must specify
    - the computation and how it is to be partitioned among processors,
    - the synchronization and data movement needed to ensure that the program computes the correct answers and achieves high performance.
- the nature of high-end computing systems changes rapidly =>
  - must be possible to express programs in a reasonably machine-independent way,
  - In other words, parallel programs should be portable between different architectures.
  - this is a difficult ideal to achieve because the price of portability is often performance

Complexity of the problem

- A complicating factor for parallel computing
- This complexity requires
  - extraordinary skill on the part of the application developer
  - extraordinary flexibility in the developed applications.
- Often this means that parallel programs will be developed using multiple programming paradigms and often multiple languages.
- Interoperability is thus an important consideration in choosing the development language for a particular application component.

Portability is elusive

- At the beginning of parallel computing era, every vendor of parallel systems offered a different API
  - This made it extremely difficult for developers of parallel applications (work repeated for each new parallel architecture).
- The Message Passing Interface (MPI) standard
- Portability is not just a matter of implementing a standard interface: most users are interested in *portable performance:*
  - the ability to achieve a high fraction of the performance possible on each machine from the same program image.
  - implementations of standard interfaces are not the same on each platform, portability, even for programs written in MPI, has not been automatically achieved.
  - The implementor must spend significant amounts of time tuning an application for each new platform.

Algorithms are not always portable

- An algorithm does not always work well on every machine architecture.
- The differences arise because of
  - the number and granularity of processors,
  - connectivity and bandwidth, and
  - the performance of the memory hierarchy on each individual processor.
- Portable algorithm libraries must be parameterized to do algorithm selection based on the architecture on which the individual routines are to run

Parallelism isn't everything

- The principal problem on scalable machines, other than parallelism, is data movement.
- Second problem: the bandwidth from main memory of shared-memory multi-processors
- Algorithms and software have had to increasingly deal with memory hierarchy issues, which are now fundamental to parallel programming.

Community acceptance

- Technical excellence alone cannot guarantee that a new software approach will be successful.
- To achieve widespread use, there has to be the expectation that a software system will survive the test of time.
- Standards are an important part of this, but cannot alone guarantee success.
- A case in point is HPF.
  - HPF failed to achieve the level of acceptance of MPI because the commercial compilers did not mature in time to gain the confidence of the community.
  - OpenMP has succeeded because it targets the market, while HPF was focused on the high end.

Performance of parallel systems

Performance Metrics for Parallel Systs.

1. Basic raw performance
2. Machine efficiency
3. Hockney's and Jesshope's model for vector processing
4. Performance measurements and Top 500

Basic raw performance - Flops

- The processing speed of computers involved in scientific calculations is usually expressed in terms of a no. of **floating-point operations completed per second**,
  - a measure used to describe the computational power of the world's largest supercomputers.
- For a long time, the basic measure was Mflops expressed as: $r = N / t$ Mflops where N represents a number of floating-point operations executed in t seconds.
  - When N floating-point operations is executed with an average speed of r Mflops, the execution time of a given algorithm can be expressed as: $t = N/r$.
- The Mflop measure has been superseded by higher-order measures:
  - Gflops (gigaflops),
  - Tflops (teraflops),
  - Pflops (petaflops) = 10^{15} floating-point operations per second.
- The floating-point operations rate can be used to characterize an algorithm executing on a given machine independently of the particular characteristics of the hardware, on which the algorithm is executed, as well as to describe the hardware itself.

Basic raw performance – Peak performance

- Many vendors of parallel computers advertise a theoretical peak performance for their machines
- This is the maximum speed with which any algorithm can be potentially executed on their hardware.
- In computational practice (outside of special simplified cases, such as matrix multiplication), this performance is unattainable.
- At the same time, however, it indicates what performance can potentially be expected from a given machine.

Basic raw performance – Benchmarks

- There are several industry standard **benchmark programs**
  - Such as Whetstone, ScaLAPACK and LINPACK benchmarks.
  - Used extensively in all advanced computer system evaluations.
- Specific benchmarks have been developed to evaluate shared, distributed, and hybrid memory parallel computers: - vary from:
  - simple parallel loops to measure the ability of parallelizing compilers
  - PERFECT benchmark which consists of thirteen programs including several fluid dynamics programs,
  - Genesis consisting of FFTs, partial differential equations, molecular dynamics, and linear algebra.
  - Particularly popular set of benchmarks developed at NASA:
    - the NAS parallel benchmarks: a suite of 8 programs with 3 versions for serial, machine-dependent, and MPI-based implementations
    - http://www.nas.nasa.gov/Software/NPB.

Machine efficiency – theoretical performance

- The runtime measurements described above give an absolute measurement of the computation time used by the program.
  - interesting to know how efficient these computations actually are!
- The **efficiency** is the ratio between the actual performance and the theoretical performance of a system.
- The **theoretical performance of a superscalar computer** is calculated as follows: $R_{peak} = n_{cores} \cdot n_{FPU} \cdot f$, where
  - n_{cores} is the number of computing cores of the computer,
  - n_{FPU} is the number of floating-point units per core, and
  - f is the clock frequency.
  - Example: for a Pentium D 830 with 2 cores, 2 FPUs per core, clock frequency 3 GHz, $R_{peak} = 2 \cdot 2 \cdot 3 \cdot 10^9$ FLOPS = 12 GFLOPS

Machine efficiency – practical performance

- For a dense matrix-matrix operation (m x k vs. k x n matrix) the number of floating-point operations required is $n_{flop} = 2mnk$,
  - Performance can then by calculated by $R = n_{flop}$/wall_time,
  - Efficiency by calculating the ratio R/R_{peak}.
- For Intel CPUs, Intel provides a performance measurement tool called VTUNE,
  - free of free of charge for non-commercial purposes on Linux
  - offers the functionality of a profiler,
  - can also measure the no. of integer and floating point operations during a program call
- Many supercomputers come with integrated counters for measuring performance.
  - These make it very simple to assess performance.

Hockney's & Jesshope's Model for Vector Processing

- Actual model?
  - applicability has been revived with the development of the Earth Simulator, which was built by the NEC Corporation out of proprietary vector processors.
  - most modern processors consist of multiple pipelines
    - performance of each such pipeline can be conceptualized in terms of the vector performance model
- Performance r_N of a vector-processing loop of length N can be expressed in terms of two parameters:
  1. r_∞ - represents the performance in Mflops for a very long loop
  2. $n_{1/2}$ the loop length for which a performance of about $r_\infty/2$ is achieved.

$r_N = r_\infty / (n_{1/2}/N + 1)$ Mflops.

Hockney's & Jesshope's Model for Vector Processing

- Example: vector update
  - in the form $x \leftarrow x + \alpha y$ (the AXPY operation)
  - can be expressed as a loop of the length N in which each repetition consists of two floating-point operations.
  - execution time:

    $T_{AXPY}(N) = 2N / 10^{6,}\ r_N = 2 \times 10^{-6}\ (n_{1/2} + N) / r_\infty$ secs.
- This model can be applied not only to predict the execution time of vectorized programs but also to develop optimal vector algorithms.
  - several algorithms (e.g. *divide-and-conquer* algs) consist of loops of different lengths, which are related to each other and can be treated as parameters of a vectorized program.

Performance Measurements and Top 500

- Performance of parallel computers – comments:
  - there is no universal yardstick to measure it,
  - the use of a single number to characterize performance such as the peak performance quoted by the manufacturer is often misleading.
  - It is common to evaluate performance in terms of benchmark runs consisting of kernels, algs, & appls so that different aspects of the computer system are measured.
    - Still dependent on the quality of software rather than just hardware characteristics.
  - The controversy over performance evaluation methods has been recognized by the CS community & there have been several recent attempts to provide objective performance metrics for parallel computers.
- Good basis for performance evaluation is provided in the Top500 list:
  - Performance on a LINPACK benchmark –
    - measure used to rank the computers
    - code that solves a sys.of linear eqs, using the best software for each platform.

Top 500

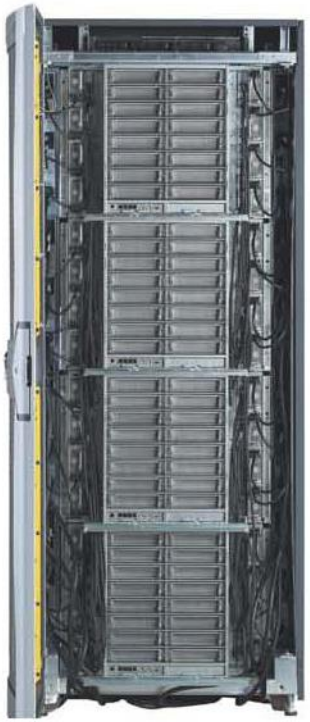
- 2004:
  - First: Yokohama *Earth Simulator* that has 5120 SX-6 vector processors from NEC Corporation.
  - Top entries rely on large numbers of commodity processors:
    - 65536 IBM PowerPC 440 processors at Livermore National Laboratory;
    - 40960 IBM PowerPC processors at the IBM Research Laboratory in Yorktown Heights;
    - 10160 Intel Itanium II processors connected by an Infiniband Network and constructed by Silicon Graphics, Inc. at the NASA Ames Research Centre.
- 2009:
  - PETAFlop/s - 10^{15} floating point operations per second supercomputer is currently available

11/2008

Cray XT5

Blue Gene

SGI Altix ICE



Cluster

Rank	Site	System	Cores	R_{max}	R_{peak}
1	DOE/NNSA/LANL United States	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz , Voltaire Infiniband IBM	129600	1105	1456.7
2	Oak Ridge National Laboratory United States	Cray XT5 QC 2.3 GHz Cray Inc.	150152	1059	1381.4
3	NASA/Ames Research Center/NAS United States	SGI Altix ICE 8200EX, Xeon QC 3.0/2.66 GHz SGI	51200	487.01	608.83
4	DOE/NNSA/LLNL United States	eServer Blue Gene Solution IBM	212992	478.2	596.38
5	Argonne National Laboratory United States	Blue Gene/P Solution IBM	163840	450.3	557.06
6	Texas Advanced Computing Center/Univ. of Texas United States	SunBlade x6420, Opteron QC 2.3 Ghz, Infiniband Sun Microsystems	62976	433.2	579.38
7	NERSC/LBNL United States	Cray XT4 QuadCore 2.3 GHz Cray Inc.	38642	266.3	355.51
8	Oak Ridge National Laboratory United States	Cray XT4 QuadCore 2.1 GHz Cray Inc.	30976	205	260.2
9	NNSA/Sandia National Laboratories United States	Sandia/ Cray Red Storm, XT3/4, 2.4/2.2 GHz dual/quad core Cray Inc.	38208	204.2	284
10	Shanghai Supercomputer Center China	Dawning 5000A, QC Opteron 1.9 Ghz, Infiniband, Windows HPC 2008 Dawning	30720	180.6	233.47
11	Forschungszentrum Juelich (FZJ) Germany	Blue Gene/P Solution IBM	65536	180	222.82
12	New Mexico Computing Applications Center (NMCAC) United States	SGI Altix ICE 8200, Xeon quad core 3.0 GHz SGI	14336	133.2	172.03
13	Computational Research Laboratories, TATA SONS India	Cluster Platform 3000 BL460c, Xeon 53xx 3GHz, Infiniband Hewlett-Packard	14384	132.8	172.61
14	Grand Equipement National de Calcul Intensif - Centre Informatique National de l'Enseignement Suprerieur (GENCI-CINES) France	SGI Altix ICE 8200EX, Xeon quad core 3.0 GHz SGI	12288	128.4	146.74
15	National Institute for Computational Sciences/University of Tennessee United States	Cray XT4 QuadCore 2.3 GHz Cray Inc.	17956	125.13	165.2