Goals for Internet/Grid Computation?

Do things you cannot otherwise do because of:

- Lack of Capacity
  - Large scale computations
- Cost
  - SETI
- Scale/Scope of communication
  - Internet searches
- All of the above

What is Performance for Internet/Grid Computation?

Sequential Computation

Assumptions

"uniform" access to memory, dedicated resources, completely reliable, static bounded resources.

Limitations

processor speed, memory size, i/o bandwidth, storage capacity

Metrics

Time to complete

Cost per unit result

Design Trade-offs

Store versus recompute

Algorithm complexity

What is Performance for Internet/Grid Computation?

Parallel Computation

Assumptions

uniform processors, dedicated, static reliable but bounded resources

Limitations

processor speed, memory per processor, number of processors, i/o bandwidth and storage capacity, communication latency, bandwidth and contention

Metrics

Speed-up, time to complete, relative efficiency, cost per unit result.

Design Trade-offs

Design paradigm, partitioning strategy, store versus recompute algorithm choice

What is Performance for Internet/Grid Computation?

Internet Computation

Assumptions

multiple execution hosts, dynamic environment, highly communication channels, unreliable resources, essentially unbounded resources

Limitations

access to resources, management of resources, fault-tolerance.

Metrics

time to complete, response times, cost per unit result, speed-up, relative efficiency

Design Issues

control strategy, heterogeneous resource management, design paradigm, store versus recompute, algorithm choice, fault management, adaptation/reconfiguration strategy.

New Issues

1. Discovery of Resources
2. Adaptive Resource Scheduling

Old Issues in Different Guises

1. Scheduling for Heterogeneous Resources
2. Fault-Tolerance
3. Security and malicious behaviors

Fundamental Question

What are the characteristics of applications will benefit from Internet scale execution environments?

Communication is intrinsic value.

Can consume arbitrarily large amounts of resources in small units.

Can be decomposed into many parallel tasks of variable size without unacceptable execution cost.

Can tolerate all of diversity, change and faults.

?????????

Relative Speed/Cost of Computation versus Communication versus Storage

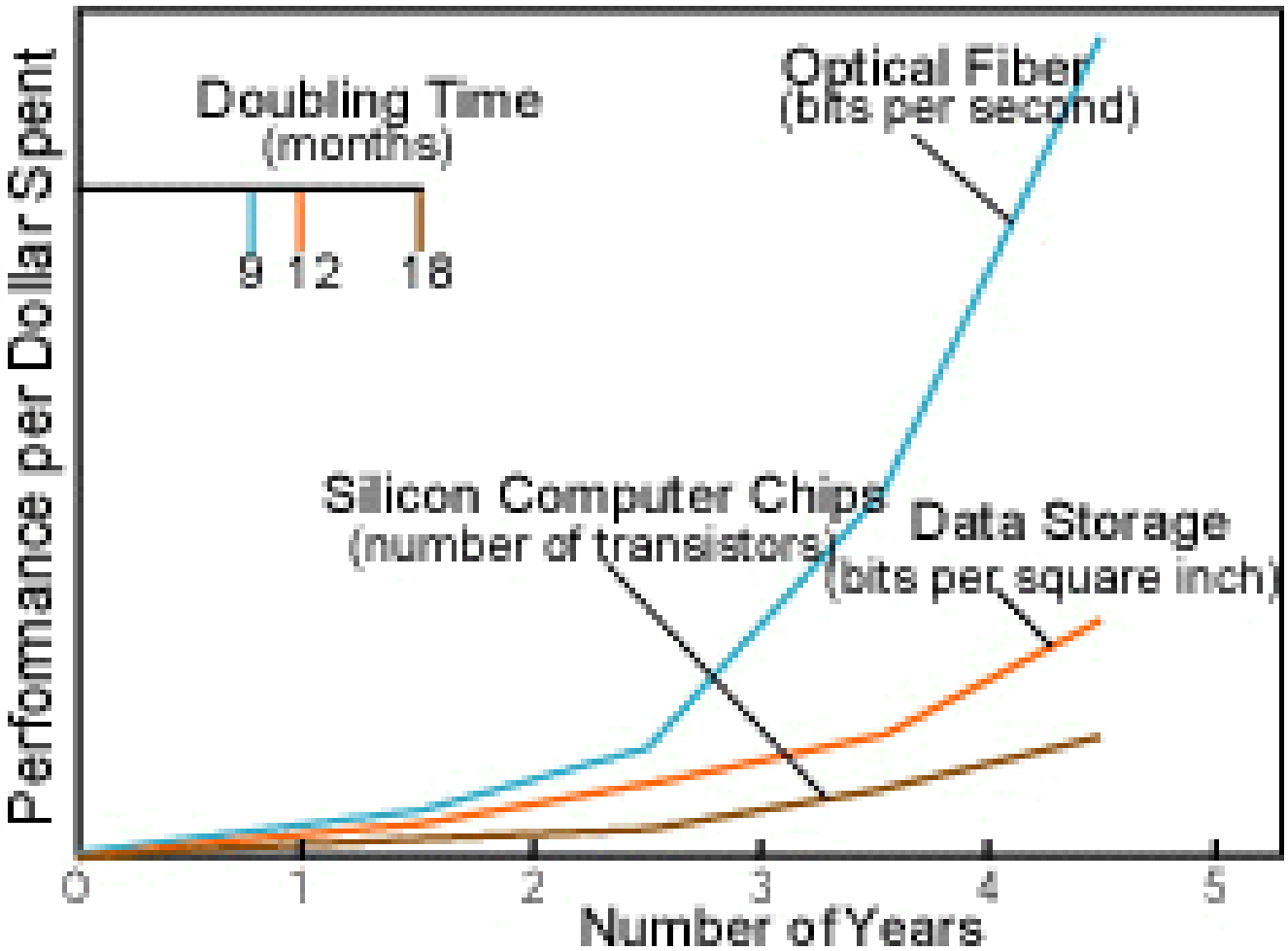
Computation power seems to be increasing at about a factor of two every two years or less at constant unit cost.

Storage size seems to be increasing at about a factor of two every two years or less at constant unit cost.

Communication Bandwidth seems to be increasing at about a factor of ???? Every ???? Years at ???? unit cost.

What is Performance for Internet/Grid Computation?

Communication Performance Of Optical Networks



Scientific American, January 2001

Quantifiable Design Issues

1. Scalable formulation paradigms
2. Speed up, performance and efficiency
   - Internet/Grid applications are intrinsically distributed parallel programs.
   - Review basics of speed up and efficiency for distributed parallel execution.
   - Consider new issues from dynamic resource sets, fault management etc.

What is Performance for Internet/Grid Computation?

Speed up for distributed parallel execution

1. Parallelizability – Amdahl's Law
2. Communication
3. Synchronization
4. Partitioning and Load Balance
   - Computation
   - Communication

Amdahl's Law – Speed Up

Assume an infinite number of processors and no communication or wait time.

Then:

Amdahl's Law states that potential program speedup is defined by the fraction of code (P) which can be parallelized:

$$\text{speedup} = \frac{1}{1 - P}$$

If none of the code can be parallelized, P = 0 and the speedup = 1 (no speedup). If all of the code is parallelized, P = 1 and the speedup is infinite (in theory).

If 50% of the code can be parallelized, maximum speedup = 2, meaning the code will run twice as fast.

Amdahl's Law - Continued

Introducing the number of processors performing the parallel fraction of work, the relationship can be modeled by:

$$\text{speedup} = \frac{1}{P/N + S}$$

where P = parallel fraction, N = number of processors and S = serial fraction.

There are limits to the scalability of parallelism. For example, at P = .50, 0.90 and 0.99 (50%, 90% and 99% of the code is parallelizable):

	speedup		
N	P = .50	P = .90	P = .99
10	1.82	5.26	9.17
100	1.98	9.17	50.25
1000	1.99	9.91	90.99
10000	1.99	9.91	99.02

What is Performance for Internet/Grid Computation?

Speed-ups are measured by using the "best" sequential algorithm executing in a single processor identical to those used for parallel execution.

Speed up is also function of the extra work which must be done when the computation is executed in parallel.

Communication

Wait or Synchronization Time

Comment: For those applications where communication is the goal, speed up is meaningless.

Communication

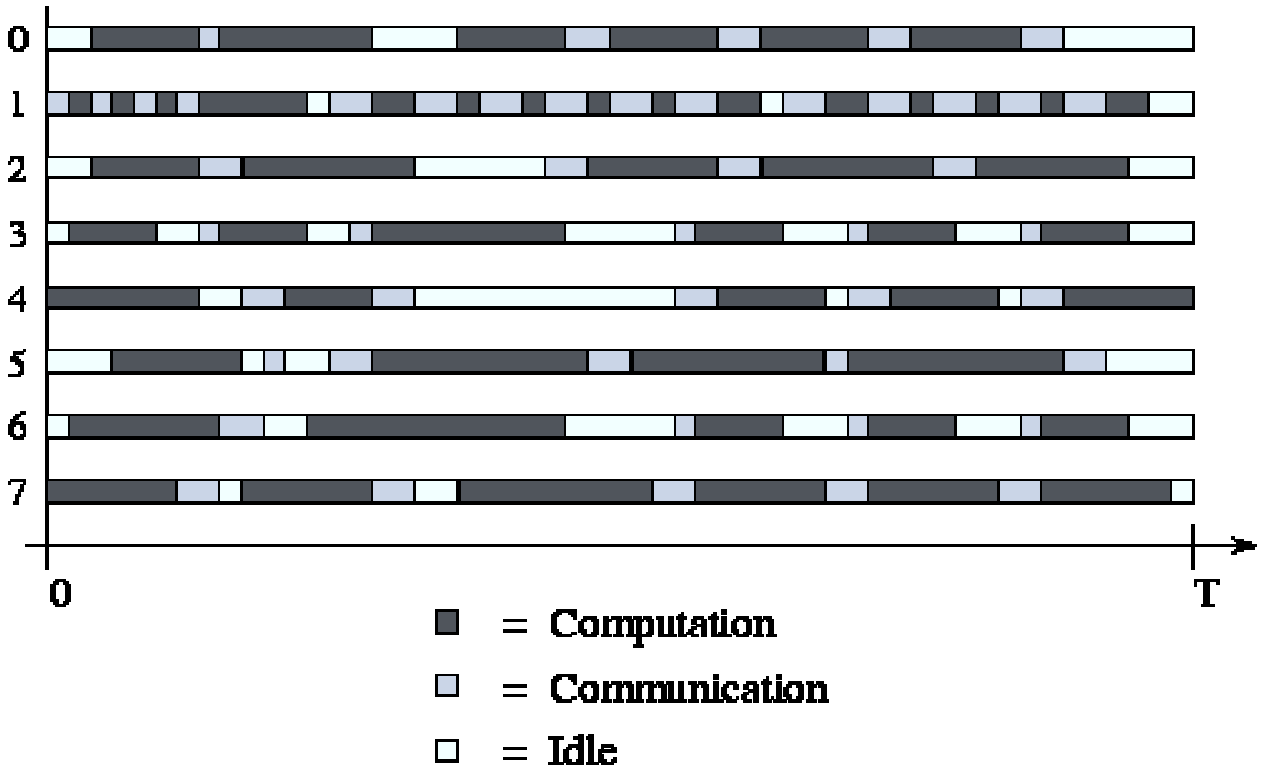
Assume that the serial fraction of the computation is zero and that all tasks require exactly the same computation time.

Then speed up is bounded by the time taken for any communication among the tasks which are executing in parallel which is not required for sequential single memory execution. Communication time can never be zero. Why?

Comment: For some applications, the task is communication and the "overhead" is the computation time necessary to implement communication.

Simple Analytic Models - Comp+Comm+Idle

Execution behavior of a program



Role of Communication Time

1. Assume the serial fraction of the computation is zero.
2. Assume that the task is decomposed into a sequence of computations followed by sending a message.
3. Assume each processor takes exactly the same time to execute the computation.
4. Assume all processors send the same data in each message for each unit computation and that the transmission time for each message is uniform.
5. Assume that the time taken to send a message is additive to the time taken for a unit computation.

Then

Speed up = $T_S(comp) / (T_P(comp) + T_P(comm))$

Speed up = T(s)/ (T(s)/N + number of messages x time for one message)

Serial fraction is never actually zero.

$T_P(comp)$ is almost never completely uniform.

Message transmission time is never quite uniform.

Simple Analytic Models for Execution Time

Let T be the time to complete. Let T_j be the time taken by the jth processor to execute its task. Let T^j_{comp} be the time take to execute the computation on processor j, etc.

$$T_j = (T^j_{comp} + T^j_{comm} + T^j_{idle})$$

$$T = \max (T_j)$$

Assume all of the processes and communication channels behave identically. Then

$$T = (1/P)\left(\sum_{i=1}^{P} T^{j}_{comp} + \sum_{i=1}^{P} T^{j}_{comm} + \sum_{i=1}^{P} T^{j}_{idle}\right)$$

$$T = (1/P)\left(T_{comp} + T_{comm} + T_{idle}\right)$$

Another metric one often hears bandied about is parallel efficiency.

$E_{relative} = T_1 / (P\ T_p)$

$E_{absolute} = T_1$ (best sequential) $/ (P\ T_p)$

Often efficiency is decreased by poor synchronization behaviors.

Scalability – What and Why?

Scalability for Fixed Problem Size

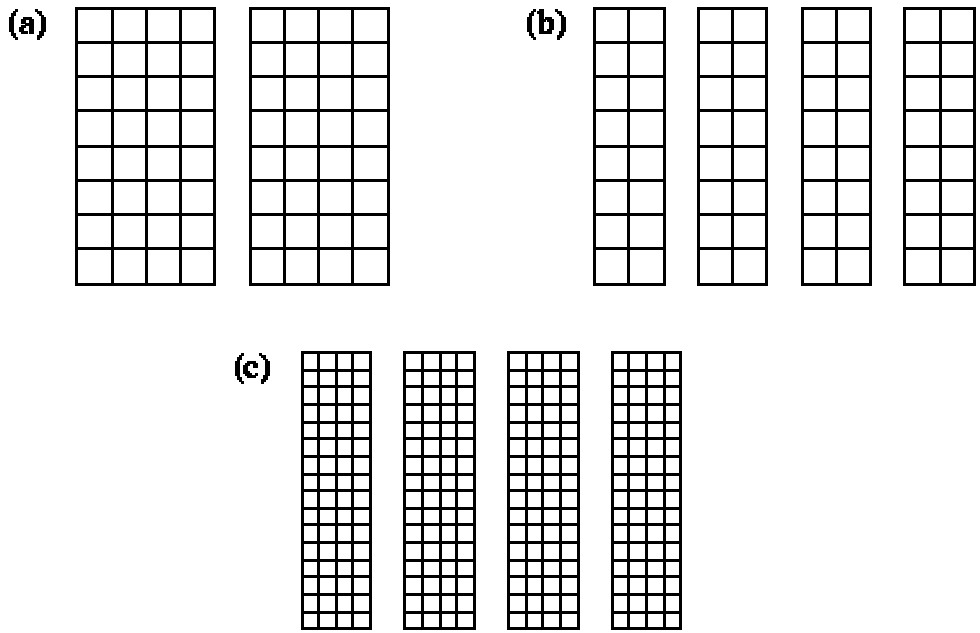
How many resources can be effectively applied to a computation of a given size? How does efficiency vary as P increases with N constant?

Scalability for Increasing Problem Size

How big a problem can I effectively solve with the what resources? What is the dependence of E on both N and P?

A computation is scalable if E is constant as P increases.

Parallel efficiency in partitioning of a 2-D grid.



Efficiency as a function of communication cost.

1-D partitioning of a 2-D grid

a) Partition 64 points among two processors.

b) Partition 64 points among 4 processors. E decreases with respect to a)

c) Partition 128 points among four processors. E is the same as for a)

Partitioning and Scalability

Example – Finite Difference computation on a three D mesh of dimension NxNxZ.

The computation is that each mesh point is the average of its neighboring mesh points.

T(sequential) ~ t_cNNZ

Partition mesh into P partitions along one-dimension.

T(comm,1D part.) = time to send one plane of mesh in each direction.

T(comm) = 2P(t_s + t_w 2NZ)

Assume P divides N and neglect idle time.

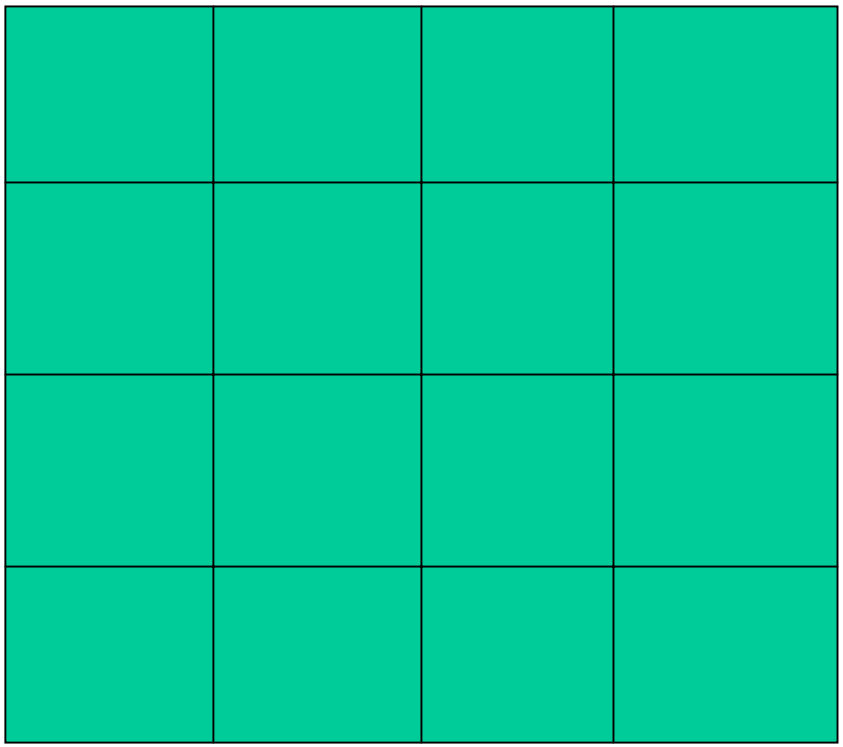
T(parallel,1D part.) = t(comp) NNZ/P + 2 t_s + t_w 2NZ

$E_{relative}$ (parallel,1D part.) = t_p NxNxZ / ($t_c N^2 tZ + 2t_s P + 4NZPt_w$)

($T_1 = t_c$ NxNxZ)

- E decreases with increasing P, t_s, and t_w. The asymptotic decrease with P is as 1/P.
- E increases with increasing N, Z and t_c.
- Execution time decreases with increasing P but has a lower bound determined by T(comm).
- Execution time increases with increasing N, Z, t_c , t_s, and t_w
- We conclude this computation as formulated is not scalable for constant N and Z.
- But let Z = N and both N and P increase to asymptotic values. Is the computation scalable?

Scalability of the Finite Difference Computation with a Two-Dimensional Partitioning of the Mesh



Each computation is of size (n/sqrt(P))(N/sqrt(P))Z

Each communication operation is of size 2(N/sqrt(P))Z

Efficiency for the computation with two dimension partitioning is

$$E_{relative}\ (parallel,2D\ part.) = t_p\ NxNxZ\ /\ (t_cN^2Z + 2t_s\ P + 4NZsqrt(P)\ t_w)$$

Let Z =N and N = SQRT(P)

$$E_{relative}\ (parallel,2D\ part.) = t_p\ PxSQRT(P)\ /\ (t_cPxSQRT(P) + 2t_s\ P + 4PxSQRT(P)\ t_w)$$

Is this computation scalable?

Isoefficiency Function

The isoefficiency function is the rate of growth of T(comp, sequential) as a function of P necessary to maintain a constant E.

T(comp,sequential) ~ NxNxZ, for

E(parallel, 1D part.) ~ t_p NxNxZ / ($t_{cZ}N^2 + 2t_s$ P + 4NxNxPt_s)
For constant E, N ~ O(P) or the computational work must grow as as O(PxP).

Or the isoefficiency function is ~ O(PxP)

While the isoefficiency function for the 2D partitioning is derived as N~O(Sqrt(P)) → isoefficiency ~O(P).