European Thematic Network 114046-CP-1-BG-ERASMUS-TN

www.cs.ucy.ac.cy/sums

PARALLEL METAHEURISTICS

PLAMENKA BOROVSKA
COMPUTER SYSTEMS DEPT.
TECHNICAL UNIVERSITY OF SOFIA

DOCTORAL EDUCATION IN COMPUTING (DEC)
Summer School on Intelligent Systems July, 2-6, 2007

THE CHALLENGE

- As the amount of data in our world grows tremendously and doubles every 24 months — the data equivalent of Moore's Law, there is a new challenge facing IT community: vast quantities of data will require new computing architectures in order to be processed.
- To ensure that today's computers are able to handle future applications, they will need to increase their processing capabilities at a rate faster than the growth of data.

REQUIREMENTS&EXPECTATIONS

- ***GREATER PERFORMANCE***
- ***LARGER MEMORY CAPACITY***
- ***INTELLIGENCE***

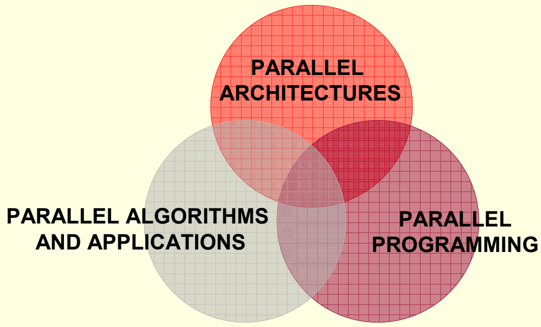
INNOVATIVE TECHNOLOGIES

- **MULTI-CORE PROCESSORS (MULTITHREADING)**
- **SCALABLE PARALLEL COMPUTER ARCHITECTURES**

  IEEE Technical Committee on Scalable Computing www.ieeetfcc.org
- **CLUSTER COMPUTING**

  IEEE Task Force on Cluster Computing www.clustercomp.org

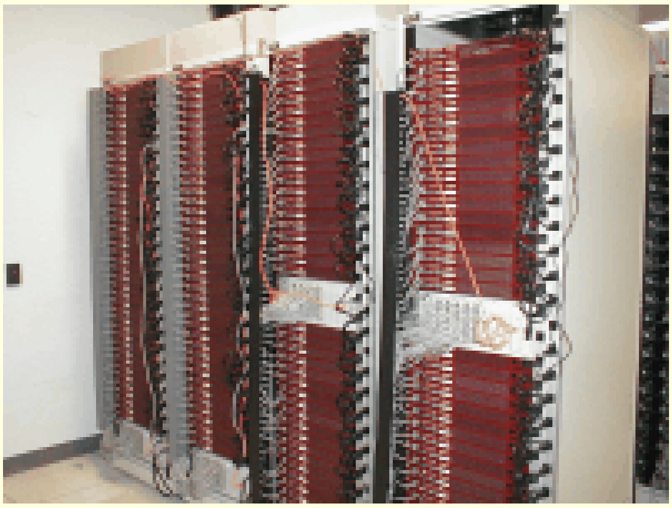
FUNDAMENTAL ASPECTS OF PARALLEL COMPUTING



Computer clusters www.topclusters.org

- Clusters of various architectural styles have become popular nowadays based on hyperthreading and multi-core processors.
- Consequently, shared memory parallel programming models are emerging as a serious competitive environment to message passing.
- Hybrid (multi-level) parallel programming model is based on a combination of the two approaches - high-level parallelism afforded by message-passing and low-level parallelism used for loop level multithreaded parallelism.

Computer Cluster Google



7

Blue Gene

8

IBM ASCI White

9

IBM ASCI Purple

10

Classification of clusters – according to 4 orthogonal attributes

Packaging:

- *Compact cluster* - nodes are packaged in 1 or more racks in a room, nodes are not attached to peripherals, called headless workstations, utilizes a high-bandwidth, low-latency communication network
- *Slack cluster* – nodes are attached to their peripherals i.e. they are complete SMP, workstations, and PCs, they may be located in different rooms, buildings, geographically remote regions

Classification of clusters

Control

- *Centralized – all the nodes are owned, controlled , managed & administered by a central operator (normally compact cluster)*
- *Decentralized – the nodes have individual owners ; the owner can reconfigure, upgrade or even shutdown the workstation at any time*

A slack cluster can be either controlled or managed in a centralized or decentralized fashion.

Homogeneity

- *a homogeneous cluster – all nodes adopt the same platform*
- *a heterogeneous cluster – nodes of different platforms, process migration not feasible*

Security

- ✓ *Exposed intracluster communication* – easy to implement, but an outside machine can access the communication paths, and thus individual nodes, using standard protocols (i.e., TCP/IP)

Disadvantages:

- *Intracluster communication is not secure*
- *Outside communications may disrupt intracluster communications in an unpredictable fashion*
- *Standard communication protocols tend to have high overhead*

- ✓ *Enclosed intracluster communication* – shielded from the outside world, a disadvantage is the lack of a standard for efficient, enclosed intracluster communication

13

Dedicated Versus Enterprise Clusters

Dedicated cluster

- Typically installed in a deskside rack in a central computer room
- Typically homogeneously configured with the same type of nodes
- Managed by a single administrator group
- Typically accessed via a front-end system
- Used as substitute for traditional mainframes or supercomputers
- Installed, used & administered as a single machine
- Executes both interactive and batch jobs
- Enhanced throughput & reduced response time

14

Enterprise cluster

- Mainly used to utilize idle resources in the nodes
- Each node is usually a SMP, workstation, or PC, with all peripherals attached
- The nodes are individually owned by multiple owners; the owner's local jobs have higher priority than enterprise jobs
- The nodes are typically geographically distributed
- Configured with heterogeneous computer nodes, connected through a low-cost Ethernet

CLUSTER DESIGN ISSUES

- *Availability support* – lots of redundancy of processors, memories, disks, I/O devices, networks, operating system images, etc.
- *Single system image* – by clustering many workstations, we get a single system that is equivalent to one huge workstation, a *megastation*
- *Job management – batching, load balancing, parallel processing*
- *Efficient communication – often use commodity networks (Ethernet, ATM) with standard communication protocols (high overheads), long wires imply larger latency, clock skew & cross-talking problems*

AVAILABILITY SUPPORT FOR CLUSTERING

RAS – RELIABILITY, AVAILABILITY, SERVICEABILITY

- ***Reliability*** – measures how long a system can operate without a breakdown
- ***Availability*** – indicates the percentage of time that a system is available to the user (the percentage of system uptime)
- ***Serviceability*** – how easy it is to service the system, including hardware & software maintenance, repair, upgrade, etc.

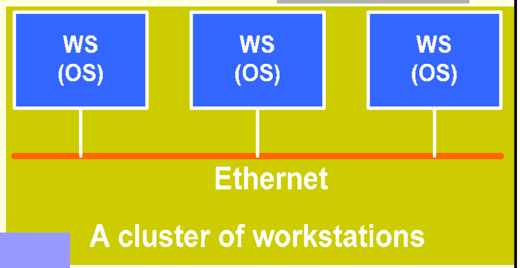
The availability concept

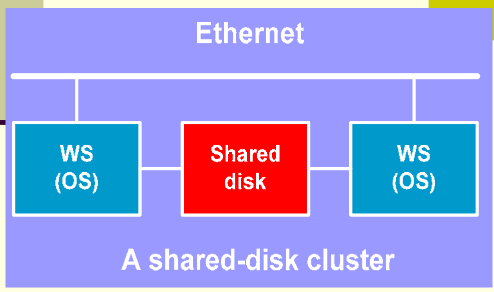
The operate-repair cycle of a computer system

- A system's reliability is measured by the mean time to failure MTTF, which is the average time of operation before the system (or a component) fails
- The metric for serviceability is the mean time to repair MTTR (the av. time to repair the system)
- **Availability = MTTF/(MTTF+MTTR)**

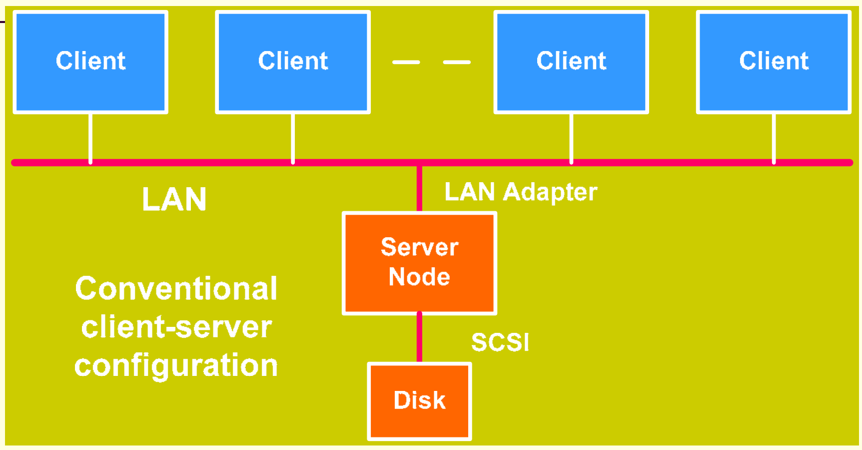
Single points of failure in clusters

Single points of failure – hardware or software components whose failure will bring down the entire system



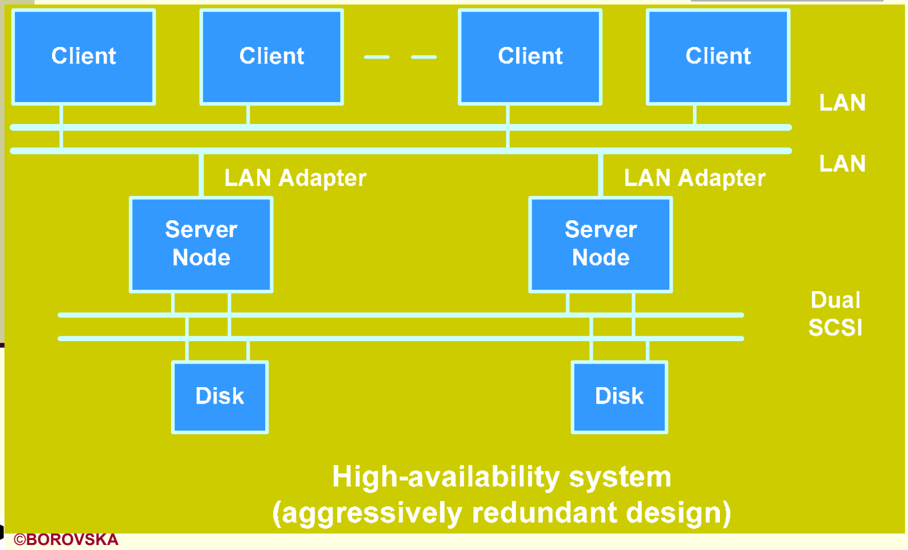


Single points of failure in clusters



5 possible single points of failure: (1)LAN network; (2)LAN adapter of the server node, (3)server; (4)SCSI bus; (5)External disk

Duplication of cluster resources to eliminate all single points of failure



Redundant components configurations

- ***Hot Standby*** – a primary component provides service, a redundant backup component is ready (hot) to take over when the primary fails (economical design – one standby component to back up multiple primary components)
- ***Mutual Takeover*** – all components are primary; 1 component fails – its workload is distributed to other components
- ***Fault-tolerant*** – N components deliver the performance of only 1 component

22

CLUSTER PRODUCTS

- Over *100 000 computer clusters* are in use worldwide
- These include both *commercial products and custom-designed clusters*
- *Nodes are mostly PCs, workstations, & SMP servers*
- The cluster sizes are mostly in the order of tens; only a few clusters exceed 100 nodes
- Most clusters use *commodity networks* such as Fast or Gigabit Ethernet, FDDI rings, ATM or Myrinet switches besides regular LAN connections among the nodes

Cluster of SMP Servers

A visible industrial trend is to cluster a number of homogeneous SMP servers together as an integrated *superserver*

SYSTEM AREA NETWORKS

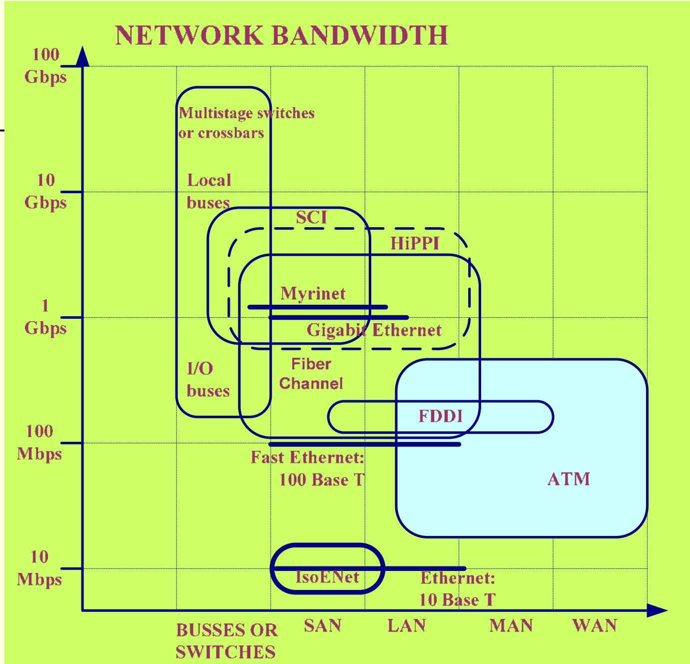
FOR PARALLEL COMPUTERS

25

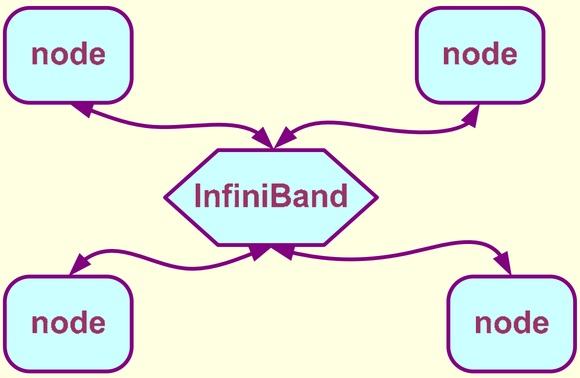
GOALS

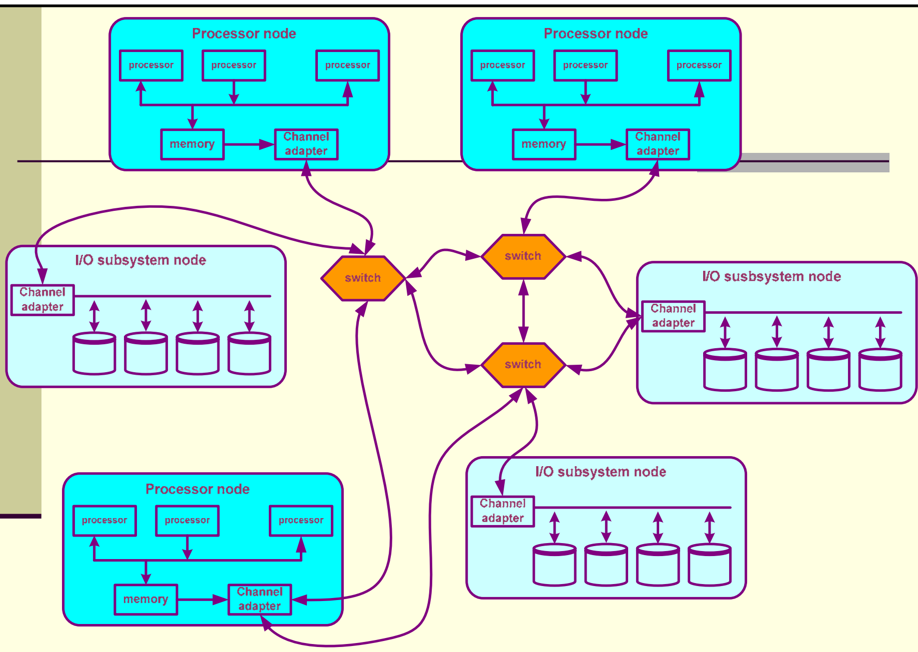
The job of an interconnection network in a parallel machine is *to transfer information from any source node to any desired destination node*, in support of the *network transactions* that are used to realize the *programming model*. It should accomplish this task with *as small a latency as possible*, and it should allow *a large number of such transfers to take place concurrently*. It should be *inexpensive* relative to the cost of the rest of the machine.

26



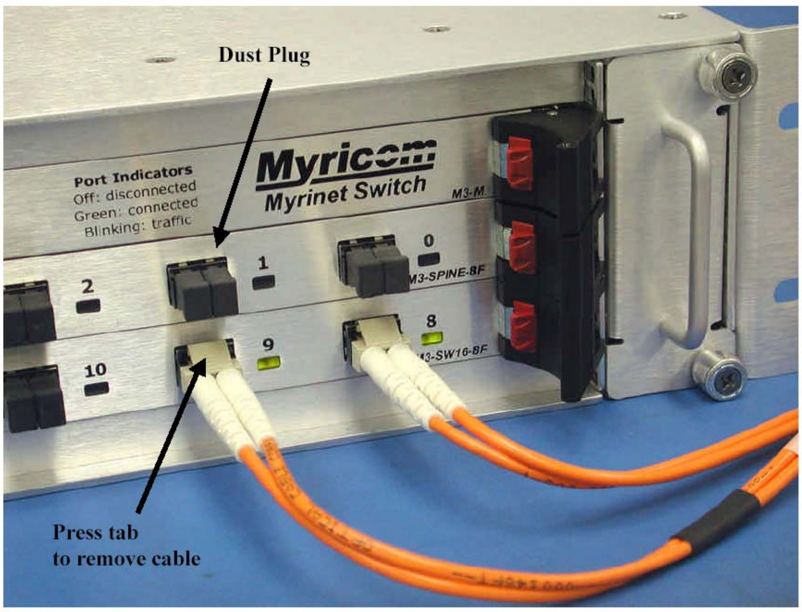
In order to overcome the limitations of the contemporary I/O system, in 2000 the Trade association InfiniBand, uniting 7 industrial leaders Compaq, Dell, Hewlett-Packard, IBM, Intel, Microsoft u Sun Microsystems created the specification of the switch architecture InfiniBand.



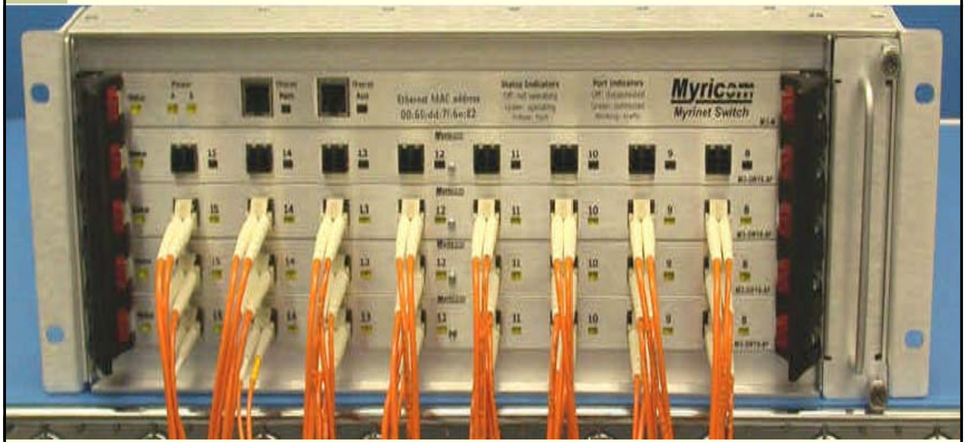


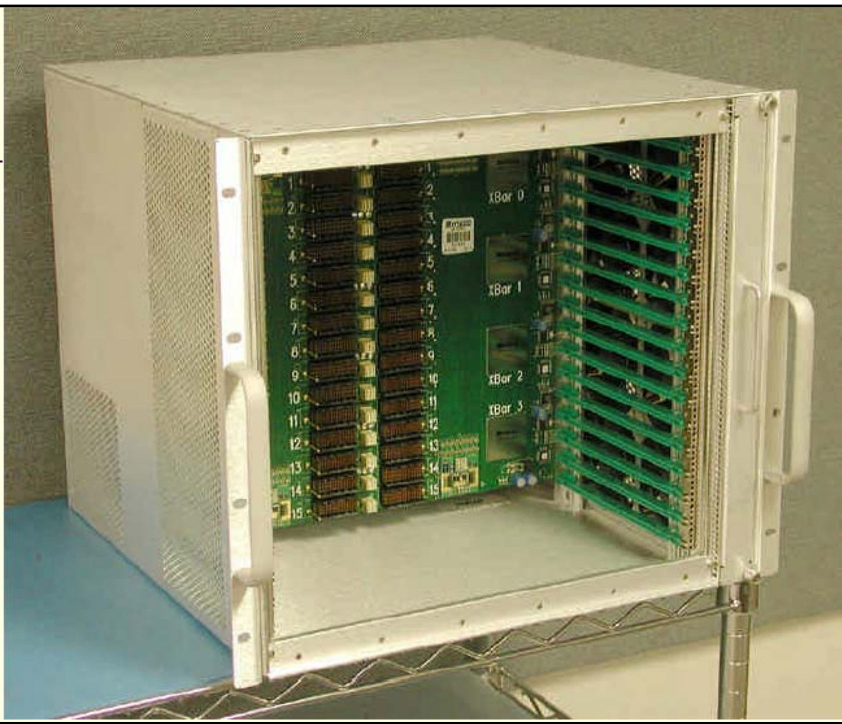
©BOROVSKA

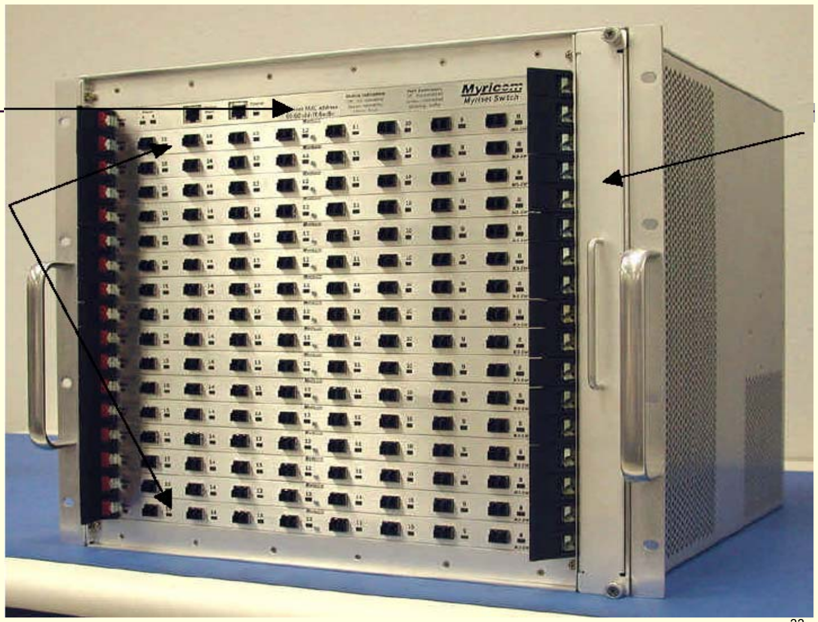
29



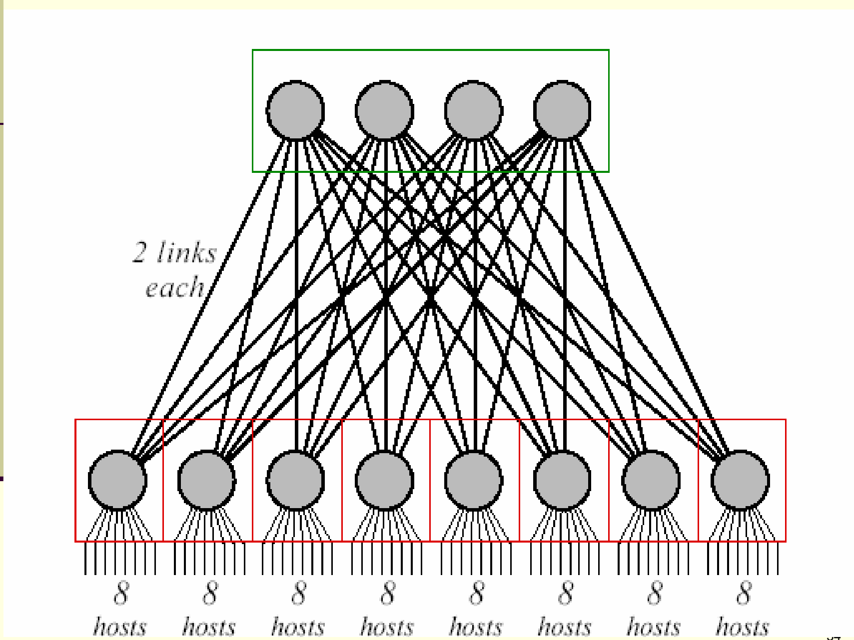
30

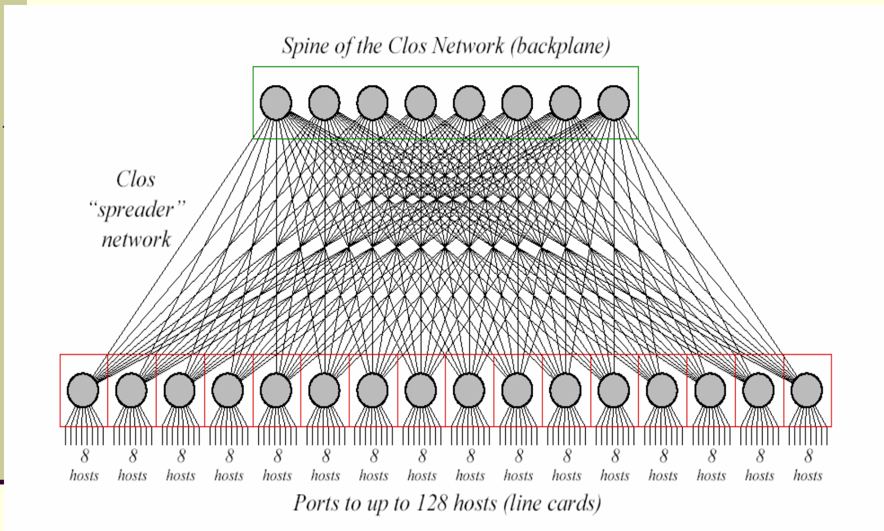

Myricom
Myrinet Switch


XBar 0
XBar 1
XBar 2
XBar 3



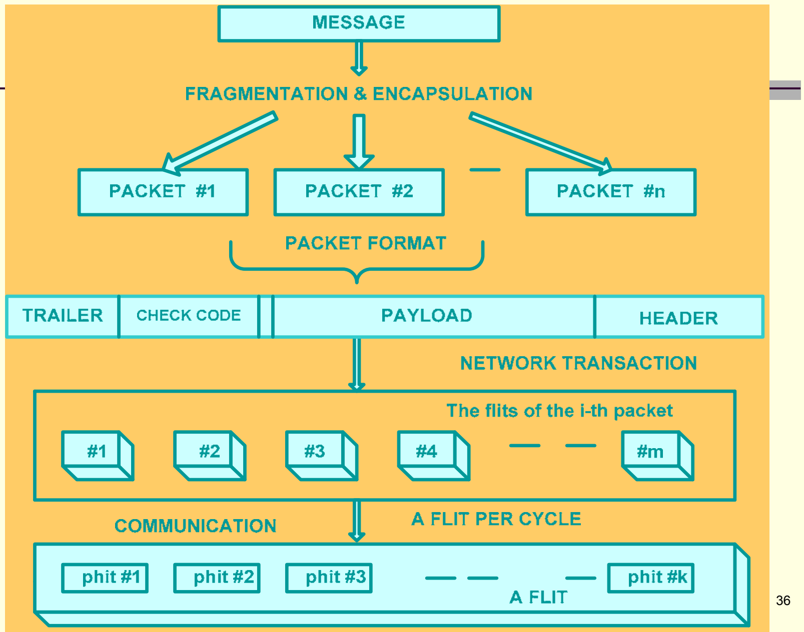
33


2 links each
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts
8 hosts



http://www.myri.com/myrinet/

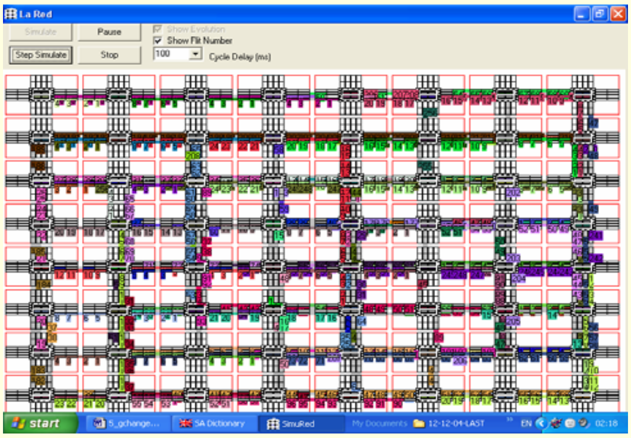
Information fragmentation and encapsulation

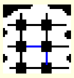


36

Communication Performance

Simured (University of Barcelona)
http://tapec.uv.es/simured/index_en.php





Simured.exe

37

Challenges of parallel computing

- The transition from sequential process of calculation to parallel computations on a parallel computer platform introduces several specifics at different stages of parallel software design and implementation.
- The latter are in close connection with the fundamental differences between sequential and parallel computations and specifics imposed by the development and use of parallel algorithms and parallel architectures.

38

Parallel programming

Parallel programming is a more complex intellectual process than writing sequential programs due to the necessity to embrace all of the aspects of the sequential programming techniques and in addition to solve new problems and meet new challenges posed by the parallelism involved at many different levels and in many different views.

39

PARALLEL PROGRAMMING ENVIRONMENTS

- **API – FLAT PARALLEL PROGRAMMING**

MPI – MESSAGE PASSING INTERFACE

MPI Forum www.mpi-forum.org

MPICH2

The MPI standard & download www.mcs.anl.gov/mpi/

- **API – MULTITHREADED PROGRAMMING**
  - **OpenMP**

The OpenMP standard & download www.openmp.org

- **HYBRID (MULTILEVEL) PARALLEL PROGRAMMING**

MPI+OpenMP

- ***In combination with Fortran, C, C++, Visual C***

40

Parallel programming with MPI

Functions MPI_ : MPI_Init (&argc, &argv)

Default communicator : MPI_COMM_WORLD

Types of communications:

- **point-to-point communication *MPI_Send*(), *MPI_Recv(), MPI_Isend(), MPI_Irecv (I – initiate)***
- **collective communication *MPI_Bcast(), MPI_Scatter()*, *MPI_Gather()* and *MPI_Reduce()***

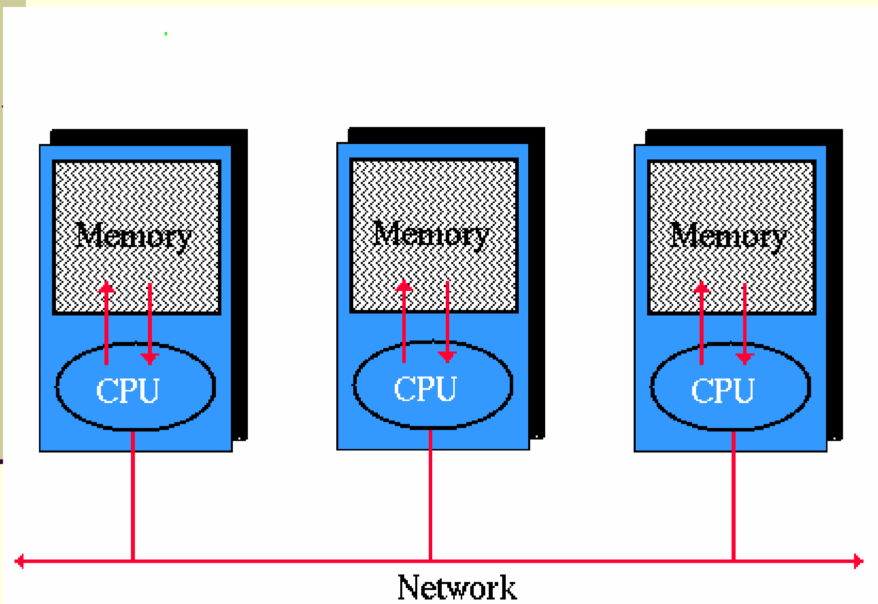
The message passing programming model

- *The standard MPI (Message Passing Interface) is the most popular specification for message passing, supporting parallel programming*
- Virtually, every commercial parallel computer supports MPI
- Libraries are available freely for use on every cluster

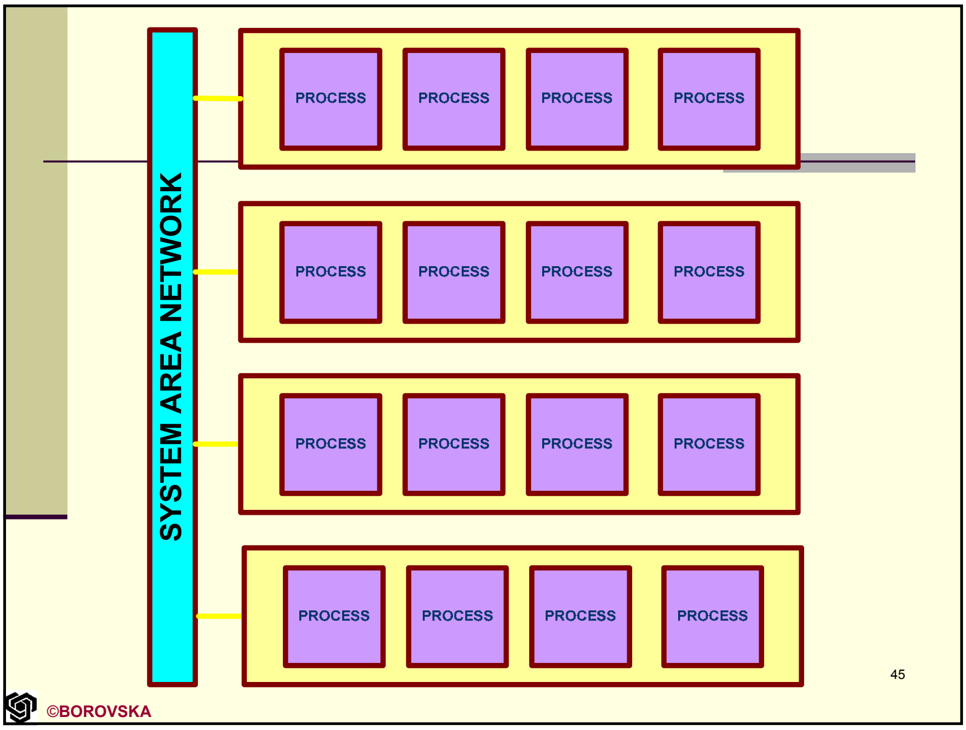
The message-passing model

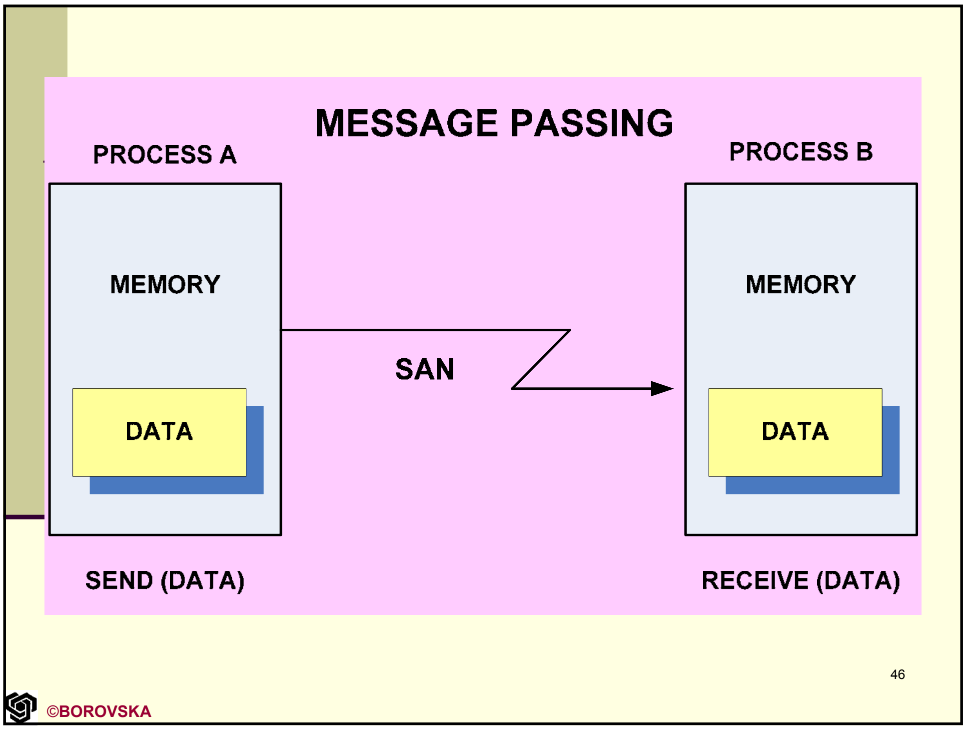
- Multiple processors, each with local memory
- SAN supports message passing between any pair of processors
- Each task of the task graph becomes a process
- Each process may communicate with any other process

43



44


SYSTEM AREA NETWORK
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
PROCESS
45
©BOROVSKA


MESSAGE PASSING
PROCESS A
PROCESS B
MEMORY
MEMORY
SAN
DATA
DATA
SEND (DATA)
RECEIVE (DATA)
46
©BOROVSKA

The message-passing model

- The number of parallel processes is fixed before starting the program
- In general, this number remains *constant* in executing the program
- Each process computes its local variables and communicates with the other processes or I/O devices, alternatively
- *Processes use communications for information exchange and synchronization, as well*

The message-passing model

- *If the message contains data → for information exchange*
- *If the message does not contain data– for synchronization*

Advantages of the message passing model

- *Efficient for wide spectrum* of MIMD *architectures*
- *Natural medium for multicomputer platforms, which do not support global address space*

The message-passing model

- Debugging programs with message passing is easier than programs with global variables
- It is easier to run deterministic programs
- Nondeterministic behavior makes debugging difficult – in various executions of the parallel program different processes acquire one and the same resources in different order

49

The message-passing model

- The first version of the message passing library **PVM (Parallel Virtual Machine)** is developed in Oak Ridge National Laboratory
- PVM – parallel program execution on heterogeneous collections of parallel and sequential machines
- 1993 г. – version 3 is published

Geist, Al, Adam Bequelin, Jack Dongarra, et all., "PVM: Parallel Virtual Machine: A User's Guide and Tutorial for Networked Parallel Computing, Cambridge, MA: The MIT Press, 1994.

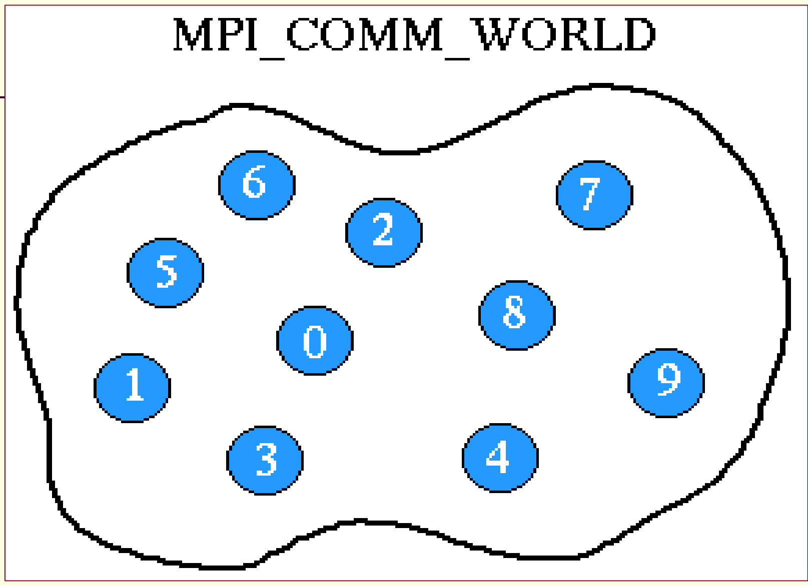
50

MPI Functions:

- *MPI_Init* – initializes MPI
- *MPI_Comm_rank* – determines the ID of a process
- *MPI_Comm_size* – determines the number of processes
- *MPI_Reduce* – reduction operation
- *MPI_Finalize* – shutdown MPI
- *MPI_Barrier* – barrier synchronization
- *MPI_Wtime* – getting the time
- *MPI_Wtick* – precision of the timer

Functions MPI_Comm_rank and MPI_Comm_size

- *After the initialization, every active process becomes a member of the communicator* **MPI_COMM_WORLD** *(default communicator)*
  - *The communicator is an opaque object, providing media for interprocess message passing*
  - *Defined communicators – processes are divided into independent communicating groups*



53

Parallel programming with MPI

- *Within the communicator processes have a strict order*
- *The position of the process in the communicator is determined by its rank*
- *In a communicator of p processes, each process has a unique rank (ID), varying from 0 to p-1*
- *A process may use its rank in order to find out for which part of the computation and data it is responsible for*

54

Parallel programming with MPI

- *A process may call function MPI_Comm_rank in order to get its rank within the communicator*
- *Function MPI_Comm_size may define the total number of processes within the communicator*

Function MPI_Finalize

After finishing all MPI calls, the process calls *MPI_Finalize* to release all system resources (for ex., the memory allocated for it)

55

Parallel programming with MPI

Compiling MPI programs

The command depends on the specific system

% mpicc –o myprog myprog.c

This command makes the system compile the MPI program in file *myprog.c* and saves the executable code in file *myprog*

Running MPI programs

% mpirun -np 2 myprog

The flag ***–np*** shows the number of processes to be created

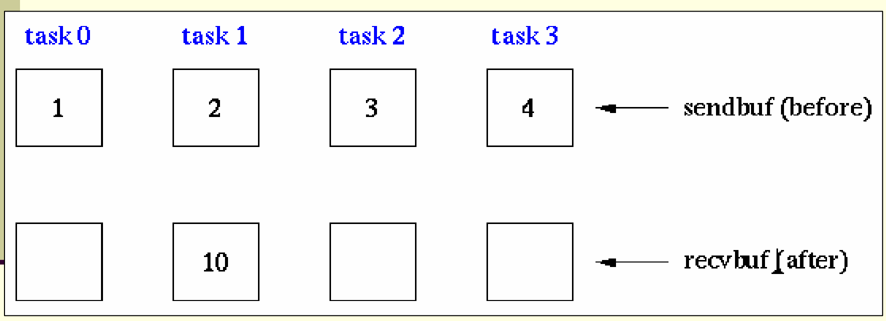
56

COLLECTIVE COMMUNICATION

- Collective communication involves a group of processes, that distribute or gather a value or multiple values
  - **Function MPI_Reduce**
- This function performs one or more reduction operations on values, sent by all the processes within the communicator

57

Function MPI_Reduce



58

Parallel programming with MPI

```
int MPI_Reduce  (
    void              *operand,   /* address of the 1st element*/
    void              *result,    /* address of the 1st result */
    int               count       /* number of reductions */
    MPI_Datatype      type,       /* type of elements */
    MPI_Op            operator,   /* reduction */
    int               root,       /* process(rank) ← result */
    MPI_Comm  comm)               /* communicator ID */
```

In spite of the fact, that only one process gets the global result, all processes within the communicator must call function MPI_Reduce, in order to take part in the collective communication!!!

59

BENCHMARKING PARALLEL PERFORMANCE

BENEFITS OF PARALLEL EXECUTION

Functions MPI_Wtime and MPI_Wtick

- **In order to measure the performance – wall clock time (seconds) – from the start until the end of program execution**
- **Generally, we ignore the time for initializing MPI processes, the communication configuration of processes and the I/O operations of the sequential devices**

60

Parallel programming with MPI

- Function MPI_Wtime returns the number of seconds, counted from a fixed time

- Function MPI_Wtick returns the precision of the result, got from MPI_Wtime.

double MPI_Wtime (void)
double MPI_Wtick (void)

We can estimate the execution time of a code section, calling twice function MPI_Wtime before and after the section. The difference of the two values, returned by the function, estimates the elapsed time in seconds.

61

Function MPI_Barrier

- **Neither of the processes can proceed after the barrier until all of the processes have reached it**
- **If the barrier is before the first call of the function MPI_Wtime, all processes will be included in the frames of the measured section approximately in one and the same time**

62

Parallel programming with MPI

Prototype of the function for barrier synchronization :

Int MPI_Barrier (MPI_Comm comm)

- **Argument shows the processes of the communicator, taking part in barrier synchronization**
- **We add local variable in function *main*:**

double elapsed_time;

- **Timer is started after initializing MPI:**

63

Parallel programming with MPI

MPI_Init (&argc, &argv);

MPI_Barrier (MPI_COMM_WORLD);

Elapsed_time = - MPIWtime ();

- After calling MPI_Reduce we can stop the timer:

MPI_Reduce (…..);

Elapsed_time + = MPI_Wtime();

As a result we can estimate the time for the execution of the parallel program

64

Parallel programming with OpenMP (Shared Memory Model)

INCREMENTAL PARALLELISM

#pragma omp <directive> [clause[[,] clause]...]

 # pragma omp <body>

#pragma omp parallel for

#pragma omp parallel sections

Clauses: ***firstprivate lastprivate***

Synchronisation: pragma critical, reduction clause

Thread scheduling: static, dynamic, guided, run time

65

©BOROVSKA

Parallel programming

- There is only one fundamental model in the sequential programming used for decades without any significant change and that is the von Neumann model while parallel programming is imposed to rapid changes and immense variety of parallel programming models and paradigms.
- Moreover the programming tools, utilities and environments, such as compilers, utilities for software tuning and debugging, profiling tools and software frameworks, are much more advanced for the sequential software development compared to the parallel programming.

66

©BOROVSKA

PARALLEL PARADIGMS

- SPMD – Single Program Multiple Data
- Manager/Workers (Task Farming)
- Divide and Conquer
- Working Pool
- Pipeline
- Phase Parallel
- Asynchronous/Synchronous Iterations

67

Abstract parallel computers

- PRAM – Parallel Random Access Machine

Fortune S., Wyllie J., Parallelism in Random Access Machine, Proceedings of ACM Symposium On Theory of Computing, pp. 114-118, 1978

- BSP – Bulk Synchronous Parallel

Valiant L., A Bridging Model for Parallel Computation, Communication of ACM, Vol. 33, pp. 103 – 111, August 1990

- Phase Parallel Model

Xu Z., Hwang K., MPP versus Clusters for Scalable Computing, Proceedings of the 2nd International Symposium on Parallel Architectures, Algorithms, and Networks, IEEE Computer Society Press, June, 1996, pp.117-123

68

PARALLEL SOFTWARE DESIGN

The parallel software developers are facing various problems that are not common for the sequential programming such as the *non-determinism* of the programs, the need for *communication and synchronization*, the definition of tasks and sub-tasks, the data distribution, *workload balance*, *deadlock and livelock*, hazards.

69

PARALLEL PROGRAMMING

- The design and development of parallel software requires ***parallel software developers and system architects to work close together***.
- Parallelism introduces additional parameters to be monitored and controlled in order to improve and optimize the parallel programs execution.

70

PARALLEL PROGRAMMING

- High performance quality and efficient implementations of parallel algorithms can be developed only if specific parallelism profiling and software tuning tools are utilized in order to monitor communication transactions execution, resources allocation, memory contents and data structures, inter-processors communications, static and dynamic processes and threads allocation.
- Usually the parallel algorithm implementations are highly dependant on the system architecture to be executed on and thus the migration form one parallel computer platform to another is not straightforward.

PARALLEL PROGRAMMING

- Decomposition of the algorithm or the data;
- Agglomeration of the tasks within processes and/or threads;
- Mapping the processes, threads and/or tasks for execution on the parallel hardware resources;
- Coordination of the parallel activities (information exchange, synchronization and mutual exclusion)
- Balance of the parallel computational workload;
- Monitoring the performance and analysis of the factors influencing the parallel system efficiency;

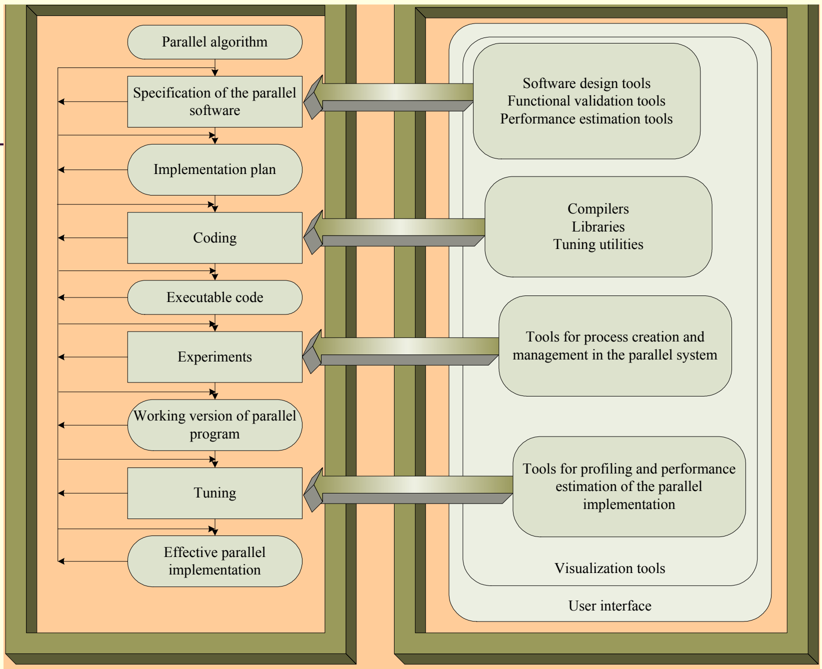
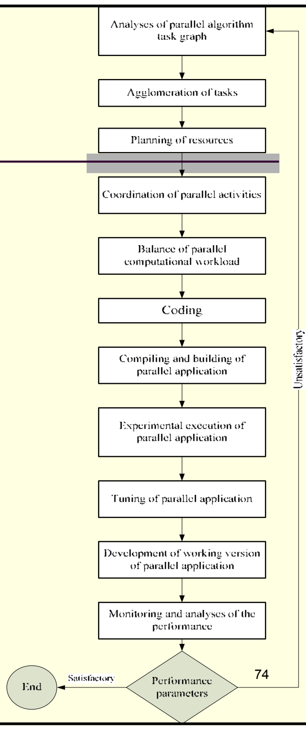


Fig. 1. The process of parallel software design and the tools utilized at each stage

- Parallel software design follows the same stages as the sequential software design:
  (1) specification of the requirements to the software, (2) design and coding, (3) tuning and testing.
- The difference between sequential and parallel software design at that stage is the necessity of thorough planning and design of communication and synchronization in the parallel program.
- The parallel program specification has to respect and contain all the required coordination of the concurrent computations.
- The parallel computation specifics require also careful analyses and estimation of parallel solution performance in respect to the features of the parallel architecture.
- The parallel program granularity and the communication structure have to be selected appropriately.



PARALLEL PERFORMANCE

- The combination of the parallel application and the parallel computer platform is called ***parallel system***

$$S_{up} = T_{seq} / T_{par}$$

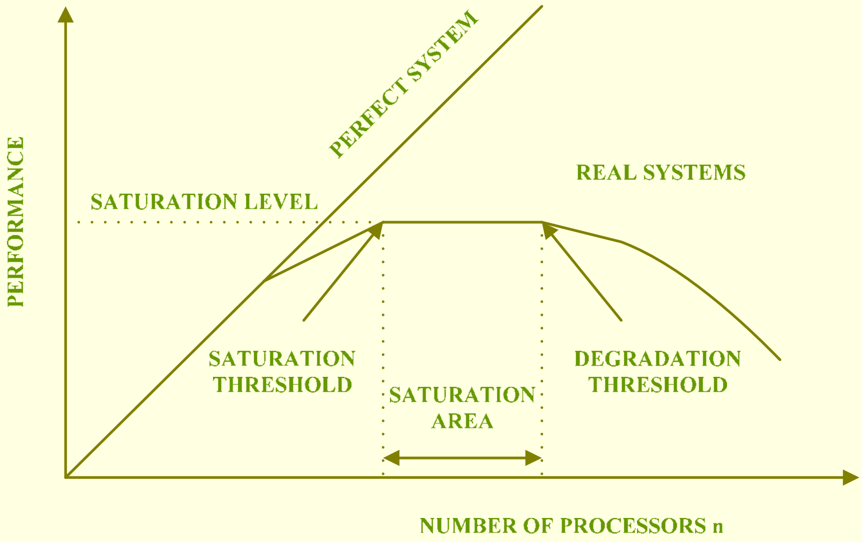
$$E_n = S_{up} / n$$

$$T_{par} = T_{comp} + T_{par} + T_{coord}$$

75

©BOROVSKA

The impact of the processors' number over parallel system's performance



ISOPERFORMANCE MODELS

- ***ISOEFFICIENCY*** – ***characterizes system scalability*** **E=f(W, n)** ; if we fix the efficiency to some constant (e.g., 50%) and solve the efficiency equation for W, the resulting function is called the ***isoefficiency function*** of the system [Grama]
- The isoefficiency metric provides *a useful tool to predict the required workload growth rate with respect to the machine size increase [Kumar]*

©BOROVSKA

77

ISOPERFORMANCE MODELS

- ***ISOSPEED*** **–** a constant speed is preserved while scaling up both machine size and problem size at the same time [Sun & Rover]

  Characterizes system scalability
- ***ISOUTILIZATION*** **–** predicts the workload growth rate with respect to the increase of machine size, while maintaining the same utilization & it is consistent with execution time i.e., a more scalable system always has a shorter execution time.

Systems with a smaller isoutilization are more scalable than those with a large one.

78

©BOROVSKA

Parallel Performance Laws

- **Amdahl's law**
  (fixed problem size)
  - **W – computational workload**
    - **α – sequential part**
  - **1- α parallelizable part**

$$S_n = \frac{W}{\alpha W + (1-\alpha)(W/n)} = \frac{n}{1+(n-1)\alpha}$$

$$S_n \to \frac{1}{\alpha} \qquad (n \to \infty)$$

$$S_n = \frac{W}{\alpha W + (1-\alpha)(W/n) + T_0} = \frac{n}{1+(n-1)\alpha + \frac{nT_0}{W}} \to \frac{1}{\alpha + \frac{T_0}{W}} (n \to \infty)$$

79

©BOROVSKA

Parallel Performance Laws

- **Gustafson's law (1988)**
  (fixed time) - scalability

W'= αW+(1- α)nW

$$S'_n = \frac{T'_s}{T'_{par}} = \frac{\alpha W + (1-\alpha)nW}{W} = \alpha + (1-\alpha)n$$

$$S'_n = \frac{T'_s}{T'_{par}} = \frac{\alpha W + (1-\alpha)nW}{W+T_0} = \frac{\alpha + (1-\alpha)n}{1+T_0/W}$$

80

©BOROVSKA

Parallel Performance Laws

Sun&Ni's Law (1993)

Memory Bound

$$S_n^* = \frac{T_s^*}{T^*_{par}} = \frac{\alpha W + (1-\alpha) G(n) W}{\alpha W + (1-\alpha) G(n) W / n} = \frac{\alpha + (1-\alpha) G(n)}{\alpha + (1-\alpha) G(n)/n}$$

$$S_n^* = \frac{\alpha W + (1-\alpha) G(n) W}{\alpha W + (1-\alpha) G(n) W / n + To} = \frac{\alpha + (1-\alpha) G(n)}{\alpha + (1-\alpha) G(n)/n + To/W}$$

- ***G(n)=1*** fixed problem size
- ***G(n)=n*** fixed time
- ***G(n)>n*** memory bound

Parameter	Taxonomy	Definition
T_s	Time for sequential processing	$T_s = \sum_{i=1}^{k} T_s(i)$
T_n	Time for parallel processing (n processors)	$T_n = \sum_{i=1}^{k} \frac{T_i(i)}{\min(L_i)} + T_0$
T_∞	Critical path	$T_\infty = \sum_{i=1}^{k} \frac{T_s(i)}{L_i}$
P_n	Parallel speed	$P_n = \frac{W}{T_n}$
S_n	Speedup	$S_n = \frac{T_s}{T_n}$
E_n	Efficiency	$E_n = \frac{T_s}{n T_n}$
U_n	Resource utilization	$U_n = \frac{P_n}{n P_{peak}}$
T_0	System overhead	$T_{0,av} = T_{par} + T_{interact}$
L_{av}	Average level of parallelism	$L_{av} = \frac{T_s}{T_\infty}$
$T_{0,av}$	Average system overhead	$T_{0,av} = \frac{T_0}{W}$
G_{av}	Average granularity	$G_{av} = \frac{W}{T_0}$

Parallel Performance Anomalies

- Normal case: 1<Sup<n
- Superlinear speedup Sup>n, E>1
- Decceleration Sup<1

83

Metaheuristics

The challenges

84

The problem area

- Solving NP (nondeterministic polynomial) problems – alternatives: exhaustive search and heuristics
- Numerous practical and theoretical tasks including applications in the areas of signal and image processing, task allocation, course scheduling, network design, graph coloring and partitioning, molecular analysis

Taxonomy

- ***Metaheuristics*** comprise high level algorithms, that coordinate the activities of heuritics algorithms, for ex. algorithms for local search, to find better quality solutions of optimization problems
- Metaheuristics include algorithms that manage the trade-off between search ***diversification***, when search is traversing bad areas of the search space, and ***intensification***, when search is focused in promising areas of the search space.

Trade-offs

- The advantages of metaheuristics include eliminating or reducing premature stops in local optima solutions that are far from the optimal solution.
- The major disadvantage is the enormous amount of computational time required.
- ***Parallel metaheuristics gives the opportunity to reduce computational time and improve the quality of solution by means of parallel search in multiple areas of the search space and utilizing various strategies for search in the search areas***.

87

Scope

- ***Genetic algorithms***
- ***Metagenetic algorithms***
- ***Memetic algorithms***
- ***Simmulated annealing***
- ***Tabu search***
- ***Ant colony optimization***
- ***GRASP – Greedy Randomized Adaptive Search Procedure***
- ***VNS – Variable Neighbor Search***

88

Genetic algorithms

- Genetic algorithms (GA's) provide search technique in computer science to find approximate solutions to optimization and search problems and present a particular class of *evolutionary algorithms*
- GA's are typically implemented as computational models simulating biological evolution.
- They perform searching a solution space for an optimal or near-to-optimal solution to a problem.

Genetic algorithms

- The possible solutions are searched by evolving populations of individuals (candidate solutions), represented by chromosomes, over multiple generations to find better solutions to the target problem.
- The evolution starts from initial population of random individuals and goes on implying the principle of "survival of the fittest".
- The genetic operations comprise activities for producing offspring such as selection, recombination and mutation.
- The fitness of the individual represents the quality of solution achieved.

Genetic operations

1. Initial population
2. Selection
3. Recombination (Crossover)
4. Mutation
5. Create Offspring
6. Evaluate fitness of individuals
7. To step 2 until convergence conditions are satisfied

Metagenetic algorithms

- Genetic algorithms of high level the purpose of which is to optimize the parameters of low level genetic algorithms
- Parameters of genetic algorithms (parameter vector): possibility of crossover, mutation rate, selection strategy, crossover strategy, etc.
- Search techniques utilizing *self-adaptation*

Simulated annealing

- Simulated annealing (SA) is a Monte Carlo based method for numerical optimization that implies the principles of thermodynamics and is motivated by an analogy to annealing in solids.
- It performs optimization without prior knowledge of the problem structure or of any particular solution strategy.

Annealing

Simulated annealing

The analogy between a physical many-particle system and a combinatorial optimization problem is based on the following equivalences:

- solutions in a combinatorial optimization problem are equivalent to states of a physical system;
- the cost of a solution is equivalent to the energy of a state;
- a control parameter plays the role of temperature, such that:
  - at high temperatures changes in energy are accepted;
  - at low temperatures only decreases or smaller increases in energy are accepted;
  - if temperature approaches zero no increases are accepted at all.

Simulated annealing

- The number of moves at each temperature is referred to as "chain length" and can be regarded as Markov chain.
- Cooling schedule of the simulated annealing is determined by the chain length and the rate at which the temperature decreases from chain to chain:

$$T_N = f(T_0, T_n, N, n)$$

where, T_0 is the initial temperature, T_N is the final temperature, and T_n is the temperature value at iteration n.

Given an appropriate cooling schedule and an infinite amount of time the algorithm will converge to the real optimum.

Memetic algorithms

Memetic algorithms are forms of genetic algorithms, that are combined with local search techniques, like simulated annealing

Tabu search

- Tabu search implies generating and evaluating mutations of individual solutions in the search space
- The individual of the least fitness is selected
- Tabu list is constructed and maintained in respect to the complete and partial solutions in order to eliminate cyclic searches and to provide searching on a large scale
- Solutions containing elements of the tabu list, are prohibited, and are being updated when moving within the search space.

Ant colony optimization (ACO)

- Multi-agents (ants) search for finding out small productive areas in the search space
- Applied for problems, where no global perspective can be analyzed and, hence, no other methods can be applied
- The quality of solution is associated with the aggregate quantity of pheromones produced by the ants

Greedy Randomized Adaptive Search Procedure (GRASP)

- Inherently, GRASP is a multistart procedure, applied successfully for finding good quality solution in solving discrete combinatorial optimization problems.
- Metaheuristics, based on the concepts of GRASP and VNS (Various Neighbor Search) requires two algorithms, the first algorithm is responsible for generating the initial solution, and the second algorithm performs the local search.

Parallel Metaheuristics - Focus

- ***Parallel Evolutionary Computing***
- ***Parallel Simulated Annealing***

101

©BOROVSKA

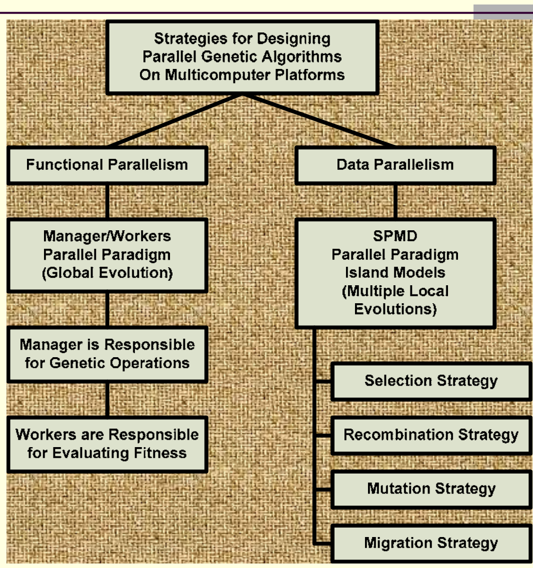
Parallel Evolutionary Computing

- The major effect of implementing parallel genetic algorithms (PGA's) is obtaining *speedup* in solving optimization problems.
- PGA's provide the opportunity to apply different strategies and techniques *to improve the convergence of the algorithm and to facilitate the quality of solution*.

102

©BOROVSKA

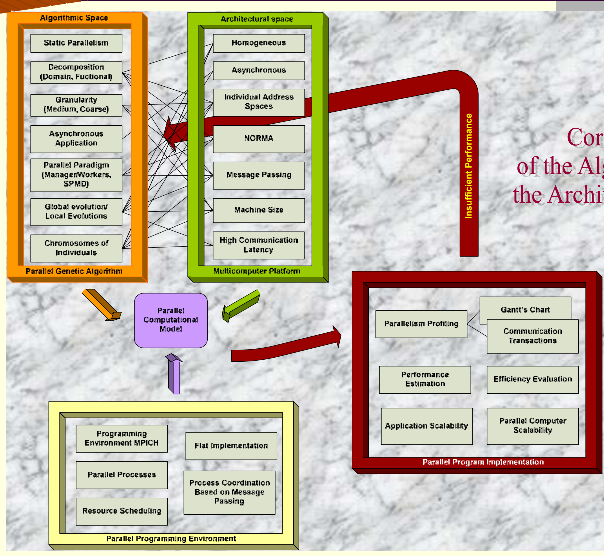
Strategies for Designing Parallel Genetic Algorithms for Multicomputer Platforms



103

©BOROVSKA

PARALLEL ALGORITHM DESIGN METHODOLOGY



Correlations
of the Algorithmic and
the Architectural Spaces

104

©BOROVSKA

TRAVELLING SALESMAN PROBLEM

- The TSP is a widely explored NP-hard combinatorial optimization problem the purpose of which is to find out the shortest tour for the salesman for a given set of cities to be visited.
- That tour is supposed to start from one city, to traverse all the cities visiting each city just once and ending at the starting city i.e. the path is cyclic. The length of that tour should be minimal.

105

THE GENETIC APPROACH

- The chromosomes of the individuals represent a tour of the cities, and fitness represents the length of the path.
- The fitness of each chromosome is calculated.
- The best individual is always selected and is not subjected to genetic operations i.e. elitism is maintained.
- A pair of chromosomes to be recombined is selected randomly.

106

THE GENETIC APPROACH

- One-point cross-over is applied – the resulting chromosome of the offspring is formed by copying the part of the chromosome of the first parent up to the point of cross-over and the second part is a copy of the chromosome of the second parent from the point of cross-over to the end of the chromosome.
- After the recombination of all pairs of chromosomes they are subjected to mutation with a specified rate – two randomly chosen genes of randomly selected chromosomes are selected and they change places i.e. two random cities change places in the routes constructed so far.
- The fitness of the newly generated offspring is calculated and the genetic operations are repeated until the convergence check is satisfactory.

107

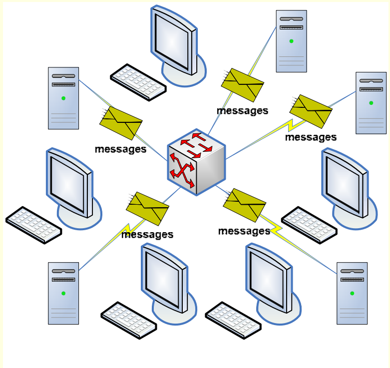
THE GENETIC APPROACH

- The genetic mutation is a random change that occurs in the characteristics of a gene.
- Mutation tends to bring about major unpredictable changes in the fitness of an individual.
- Increasing the number of mutations results in increasing the algorithm's freedom to search outside the current region of variable space.
- Mutation rate (μ) is a limitation ratio for the number of mutation events on each generation. Different mutation strategies have been applied to facilitate the convergence of PGA.
- The variation level of mutation is beneficial in parallel environments.

108

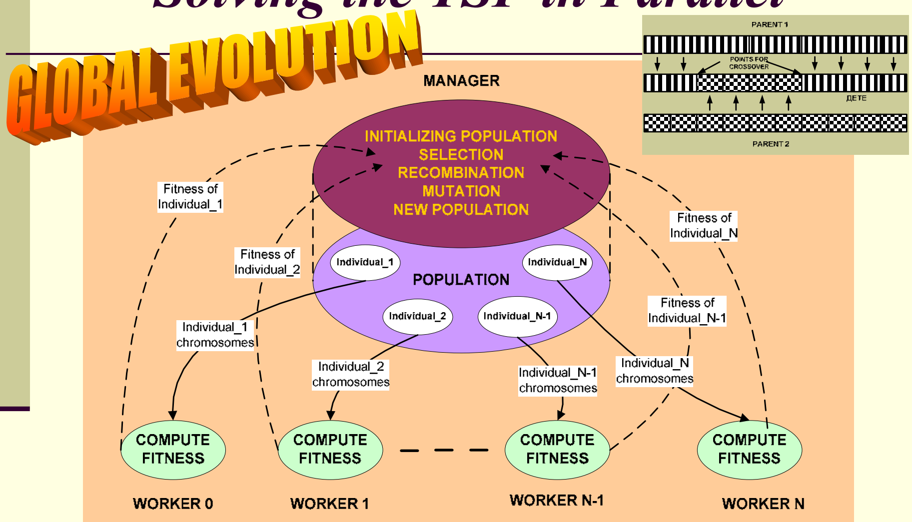
Parallel Software

- Parallel programming environment MPICH Version 1.2.4.
- Programming language compiler Microsoft Visual C++ 2005.
- Parallel profiling - Jumpshot Version 3.0.
- Parallel genetic algorithms are implemented in C++ using MPI and OpenMP.



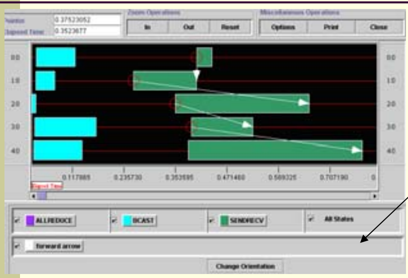
Target Computer Platform

Solving the TSP in Parallel



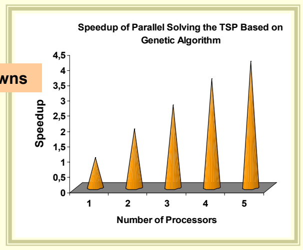
The parallel model of genetic computation
based on the manager/workers parallel paradigm

Parallelism Profiling and Benchmarking

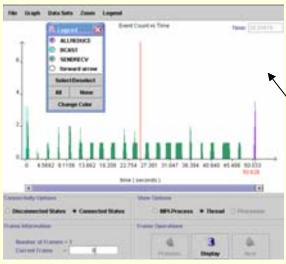


Gantt's chart
for the parallel genetic computation
of TSP

for 1000 towns

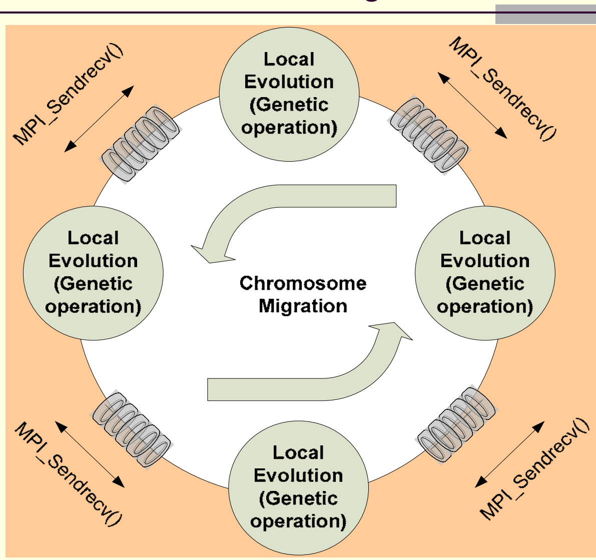


Scalability of machine size

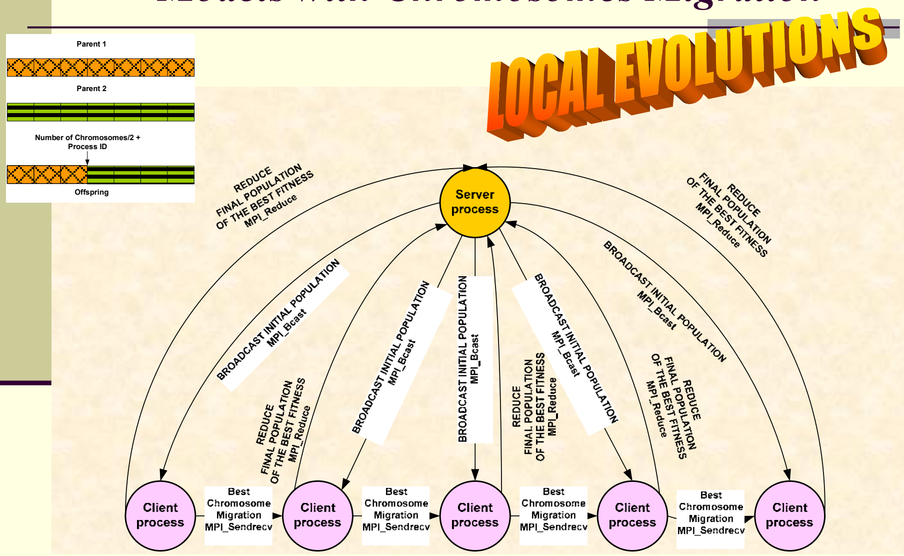


Communication Transactions Histogram

Parallel computational model for solving the TSP by genetic approach with SPMD parallel paradigm and chromosome migration



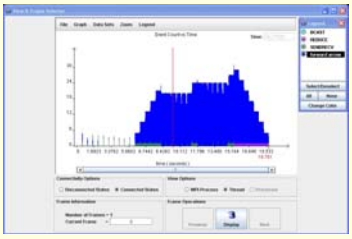
Parallel Implementations Based on Island Models with Chromosomes Migration

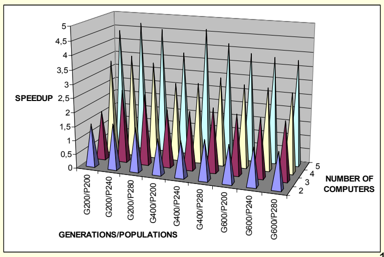


List of Experiments

Experiment #	Number of generations	Number of populations	Population size	Migration size	Mutation rate	Exchange period
1	200	200	1000	4	0.05/0.02/0.01	20
2	200	240	1200	5	0.05/0.02/0.01	20
3	200	280	1400	6	0.05/0.02/0.01	20
4	400	200	1000	4	0.05/0.02/0.01	20
5	400	240	1200	5	0.05/0.02/0.01	20
6	400	280	1400	6	0.05/0.02/0.01	20
7	600	200	1000	4	0.05/0.02/0.01	20
8	600	240	1200	5	0.05/0.02/0.01	20
9	600	280	1400	6	0.05/0.02/0.01	20

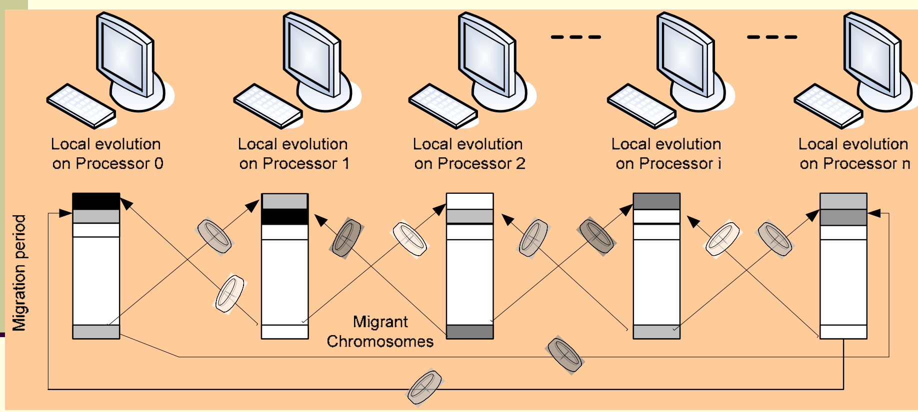
Parallelism Profiling and Benchmarking





115

Island Models with Parallel Mutation Strategies



Parallel genetic computation with periodic circular migration

116

Experimental
variable genetic parameters

#Processors	1	2	3	4	5	
#Cities	Population size	Subpopulation size				#Migrants
100	120	60	40	30	24	4
200	240	120	80	60	48	9
300	360	180	120	90	72	14
400	480	240	160	120	96	19
500	600	300	200	150	120	24
600	720	360	240	180	144	28

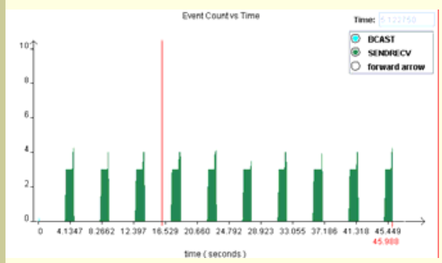
117

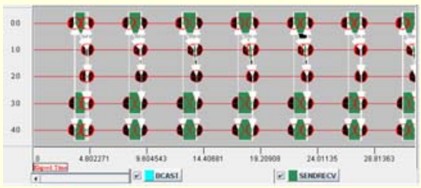
Experimental
Parallel Mutation Strategies

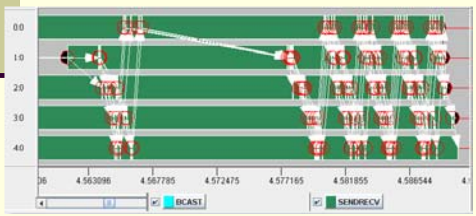
Mutation	Description	Abbreviation	Description	μ
fmr	Fixed mutation rate	Fmr0.01	Fixed mutation rate μ =0.01	μ =0.01
		Fmr0.05	Fixed mutation rate μ =0.05	μ =0.05
		Fmr0.1	Fixed mutation rate μ =0.1	μ =0.1
		Frm0.15	Fixed mutation rate μ =0.15	μ =0.15
		Fmr0.2	Fixed mutation rate μ =0.2	μ =0.2
vmrg	Variable mutation rate for each generation	vmrginc0.001	Variable mutation rate for each generation increasing $\mu+$ =0.001; $0.01 \le \mu \le 0.2$	$\mu+$ =0.001; $0.01 \le \mu \le 0.2$
		vmrgdec0.001	Variable mutation rate for each generation decreasing $\mu-$=0.001; $0.01 \le \mu \le 0.2$	$\mu-$=0.001; $0.01 \le \mu \le 0.2$
pvmr	Parallel variable mutation rate different for each process	pvmr0.05	Parallel variable mutation rate for each process has different fixed mutation rate as: μ =0.2 → P0, μ =0.15 → P1, μ =0.1 → P2, μ =0.05 → P3 and μ =0.01 → P4	$0.01 \le \mu \le 0.2$
		pvmrg±0.001	Parallel variable mutation rate for each generation on; for even processes → incrementing $\mu+$ =0.001; $0.01 \le \mu \le 0.2$ and for odd processes → decrementing $\mu-$=0.001; $0.01 \le \mu \le 0.2$	$0.01 \le \mu \le 0.2$

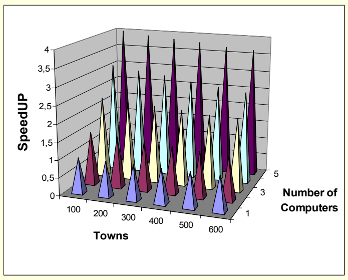
118

Parallelism Profiling and Benchmarking





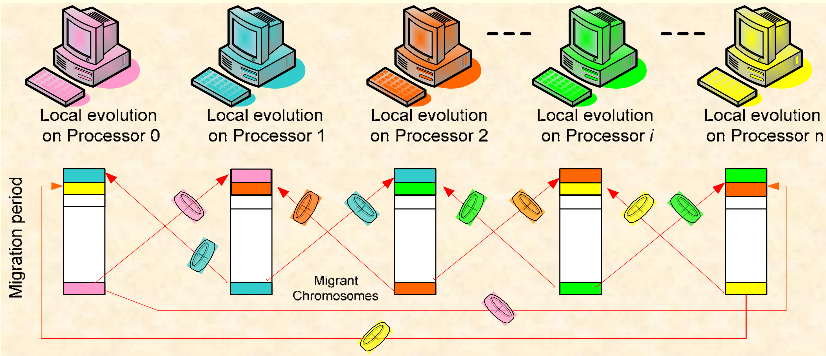




Migration Policies for Island Genetic Algorithms

- ***one way local ("slow") best chromosomes migration*** – unidirectional propagation of migrants (best individuals) to the nearest neighbor island, circular topology
- ***two way local ("fast") best chromosomes migration*** - bidirectional propagation of migrants (best individuals) to the nearest neighbor islands, circular topology
- ***global ("instant") best chromosomes migration*** – broadcast propagation of migrants (best individuals) to all islands, star topology
- ***one way local ("slow") random chromosomes migration*** – unidirectional propagation of migrants (random individuals) to the nearest neighbor island, circular topology
- ***two way local ("fast") random chromosomes migration*** - bidirectional propagation of migrants (random individuals) to the nearest neighbor islands, circular topology
- ***global ("instant") random chromosomes migration*** – broadcast propagation of best migrants (random individuals) to all islands, star topology

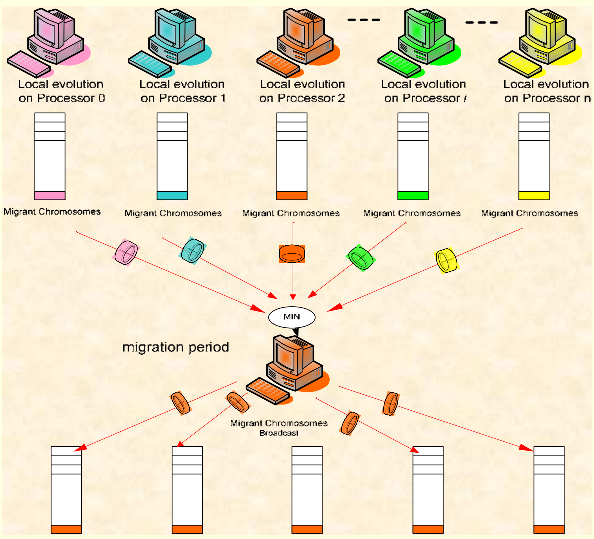
Migration Topologies for Island Genetic Algorithms



121

©BOROVSKA

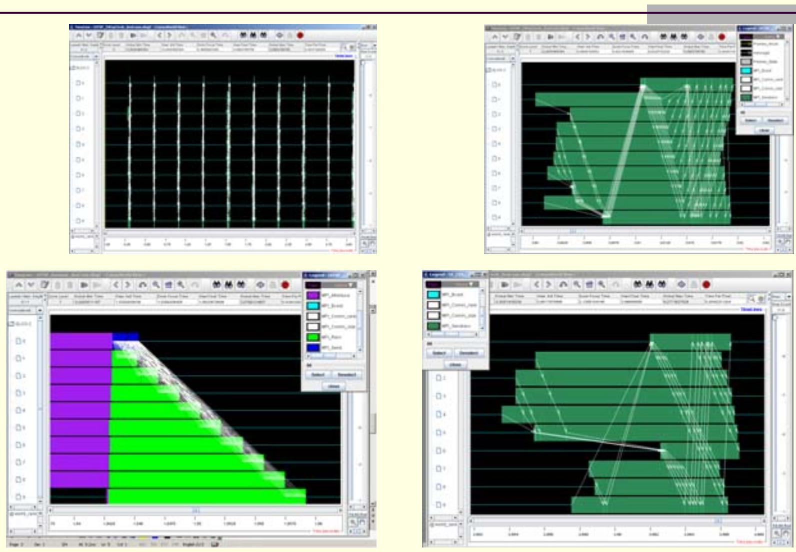
Migration Topologies for Island Genetic Algorithms



122

©BOROVSKA

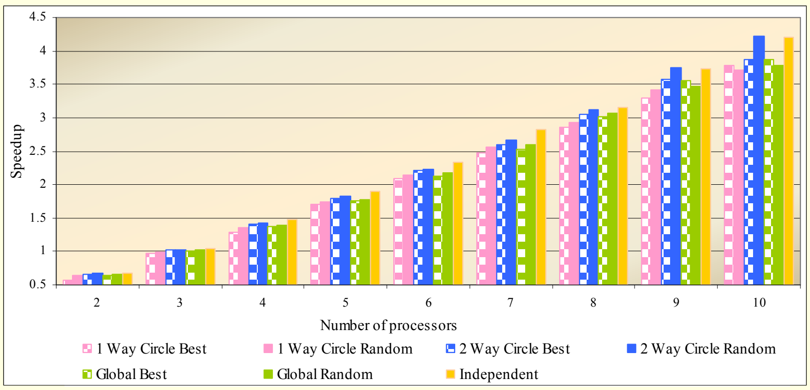
Parallelism Profiling



123

©BOROVSKA

Speedup



124

©BOROVSKA

CONCLUSIONS

- The experimental results show that the parallel coarse-granule implementation scales almost proportionally in respect to the machine size.
- The speedup slightly slows down increasing the problem size from 100 to 600 cities.

Quality of Solution Analysis

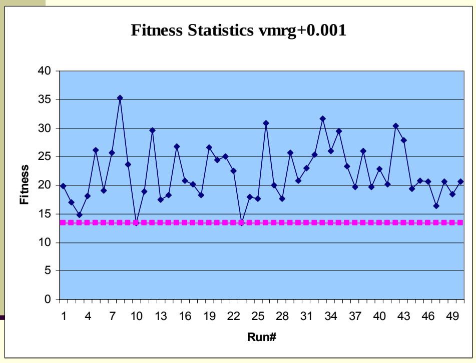
The percentage quality difference D of one run of PGA on the multicomputer platform compared to the best solution of TSP is calculated as:

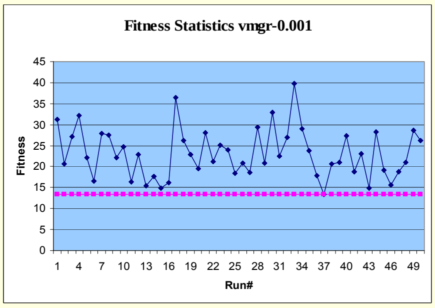
$$D = \frac{best\_fitness - perfect\_fitness}{perfect\_fitness} \times 100\%$$

$best\_fitness$ - length of the shortest tour found by the parallel genetic algorithm;

$perfect\_fitness$ -length of the shortest tour.

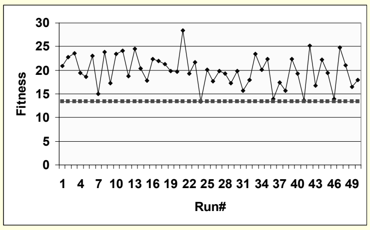
Fitness Statistics

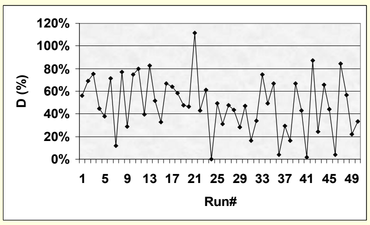


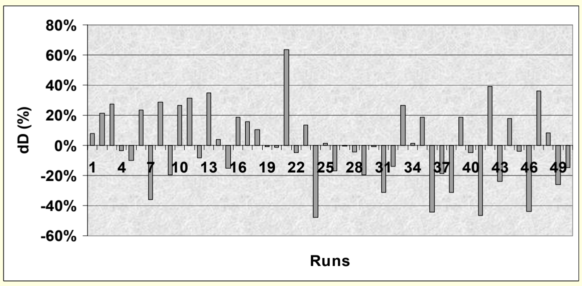


Best fitness for variable mutation rate for each generation

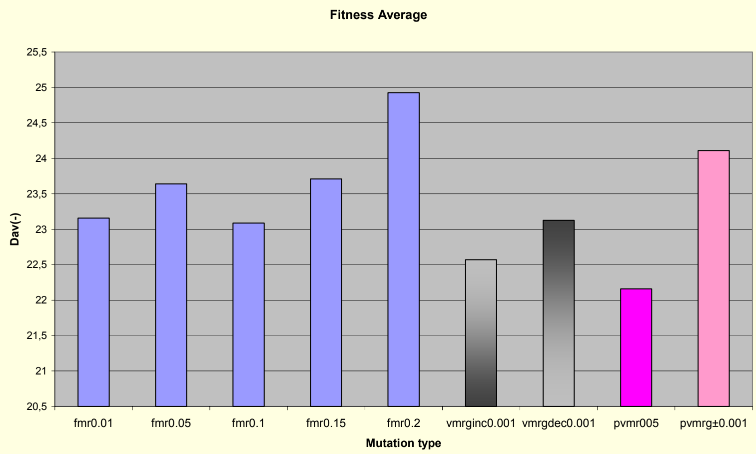
QUALITY OF SOLUTION ANALYSIS (**pvmr0.05**)







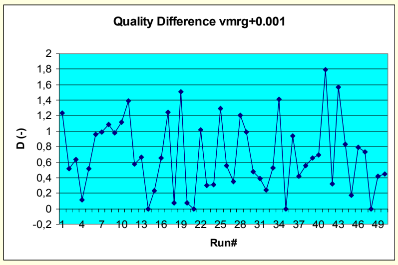
Percent Quality Difference for the Parallel Mutation Strategies

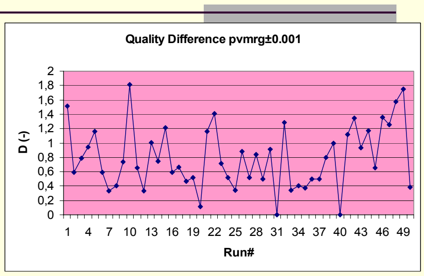


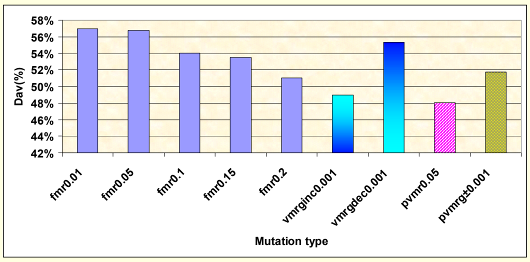
129

©BOROVSKA

Quality Difference







130

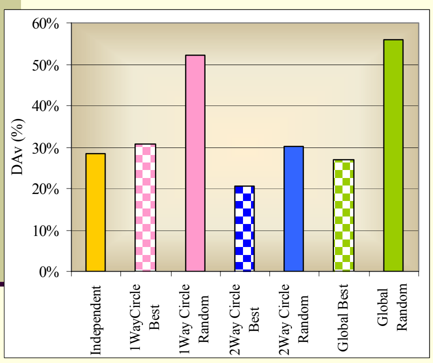
©BOROVSKA

CONCLUSIONS

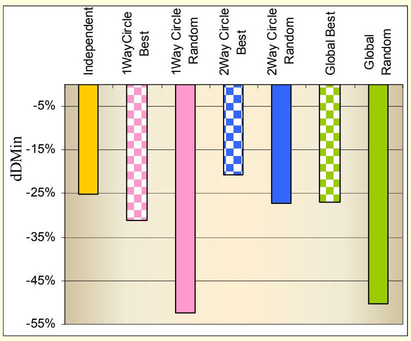
- The variation of mutation rate for each generation and for each process can improve the performance and get the best fitness of PGA.
- The best fitness achieved experimentally is for the case of parallel fixed mutation rates for each local evolution.
- This result is explained by the fact that in this case we get a great diversity of populations and a greater chance to evolve near optimal population.

131

Migration Policies



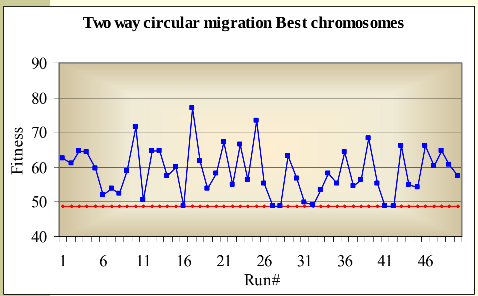
Comparison of quality difference



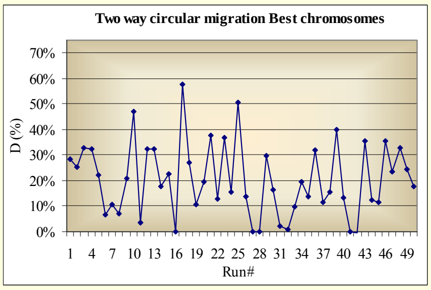
Comparison of minimum deviation

132

Migration Policies



Fitness statistics for
two-way circular
best chromosomes migration

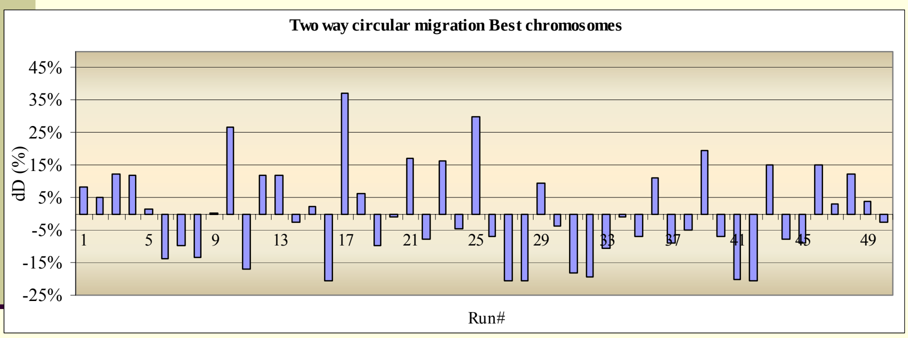


Quality difference
for two- way circular
best chromosomes migration

133

©BOROVSKA

Migration Policies



Quality difference deviation
for two-way circular migration policy
with best chromosomes migration

134

©BOROVSKA

The Room Assignment Problem

- The room assignment problem attempts to arrange N persons in N/2 rooms minimizing the cost function defined as a sum of conflicts between the room mates.
- The input data is a conflict table C of size N×N, where C[i,j] is a coefficient, denoting personal dislikes in case person i and person j share one and the same room.
- The table is symmetric with respect to the main diagonal.

135

©BOROVSKA

Simulated Annealing

- The algorithm is iterative.
- Iteration involves generating a candidate solution to the problem under investigation and estimation of its cost.
- The solution is then perturbed.
- If the perturbation results in a decrease (or no change) in the cost then it is accepted.
- If the new cost is greater, then the perturbation is accepted with a probability ,

$$e^{-\Delta t}$$

where Δ is the difference between the old and new value of the cost function and T is referred to as a temperature.

- The number of moves at each temperature is referred to as "chain length" and can be regarded as Markov chain.

136

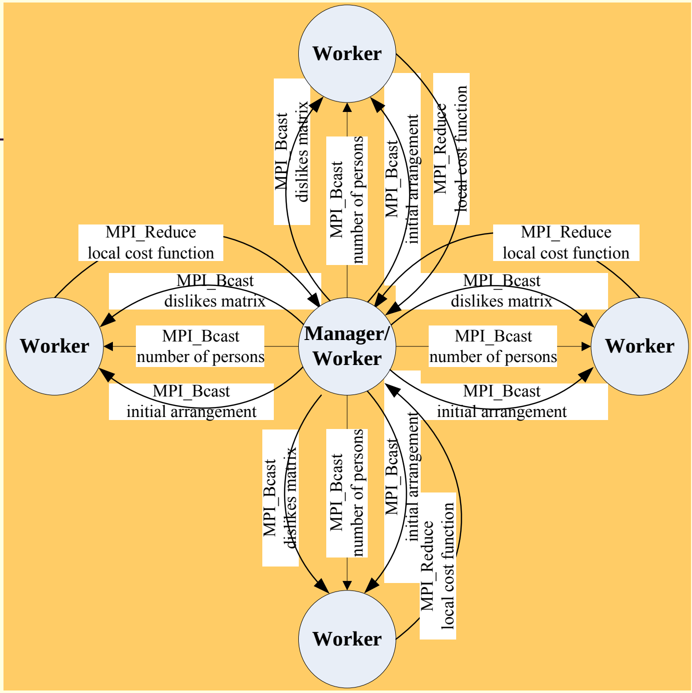
©BOROVSKA

The Room Assignment Problem

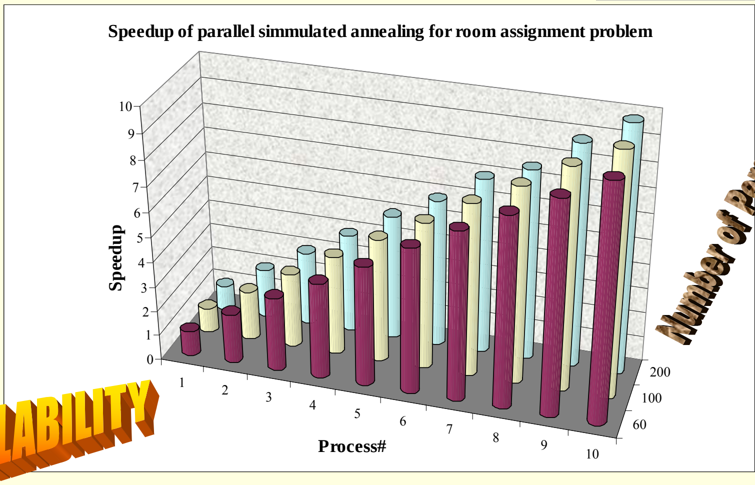
- The solution to the problem is represented as an assignment of each person to a given room;
- Persons are placed in different rooms randomly;
- The cost function is the simple sum of the dislike coefficients of all persons in the same room.
- SA operates by swapping two randomly selected persons, i.e. exchanging their rooms. The change in cost resulting from the swap is evaluated. According to the SA algorithm a decrease in cost is accepted and an increase is accepted only if it satisfies a predefined probability equation.
- In order to compute the change in cost it is not necessary to reevaluate the cost of the entire current assignment, but only the change caused by the persons that change their rooms.

137

Parallel computational model of simulated annealing for the room assignment problem



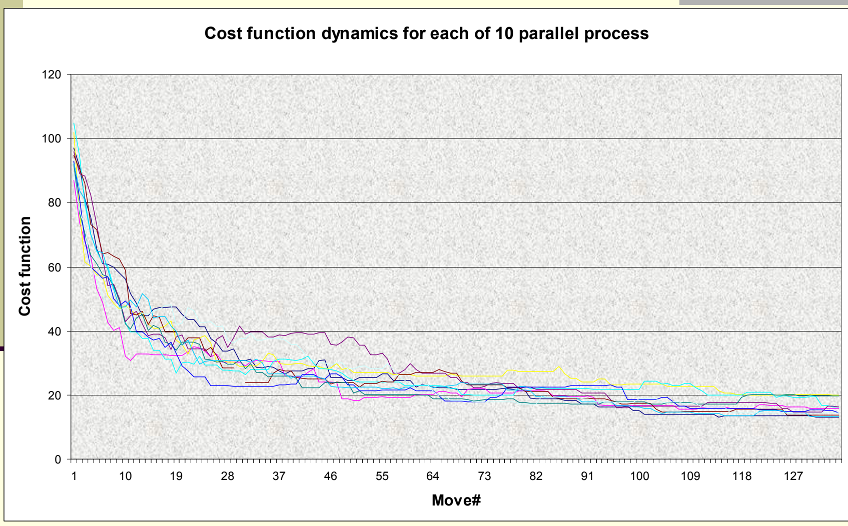
Parallel Performance



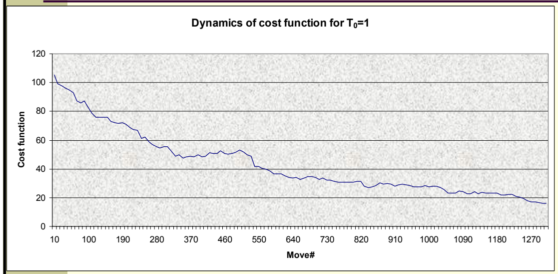
Number of Persons

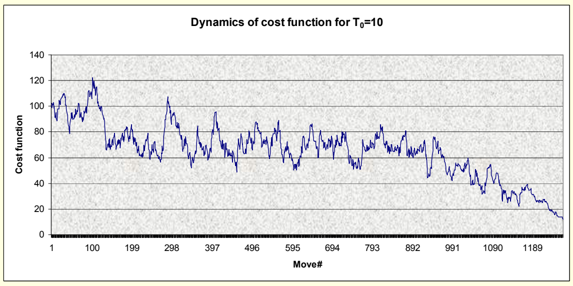
SCALABILITY

Cost Function Dynamics



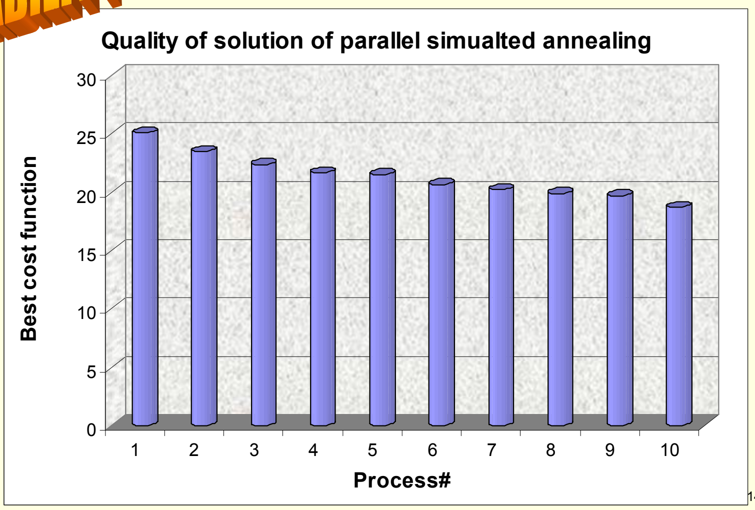
Cost Function Dynamics





Quality of Solution

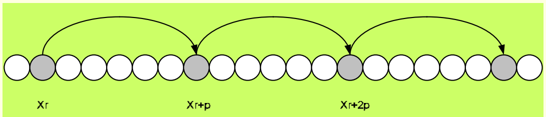
SCALABILITY



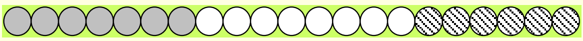
142

Parallel random generators

- Perfect RNG does not exist (no correlations of the numbers within the sequence)
- Linear congruential RNG, lagged Fibonacci RNG (C language)
- Methods for parallel RNG - manager/workers, “leapfrog”, sequence splitting, parametrization



leapfrog

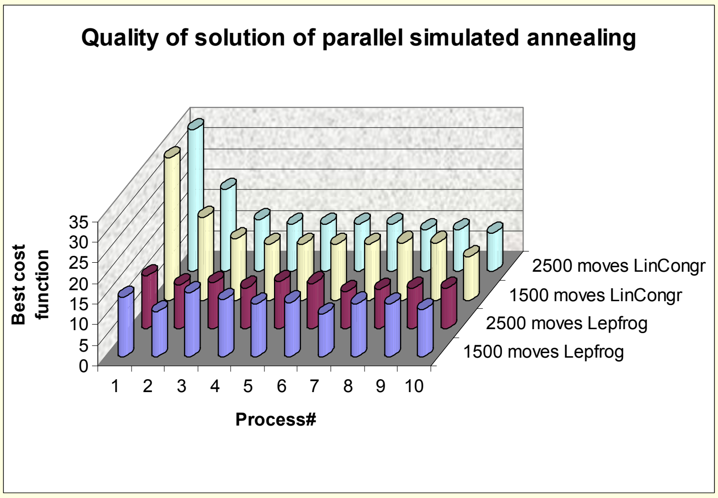


sequence splitting

143

©BOROVSKA

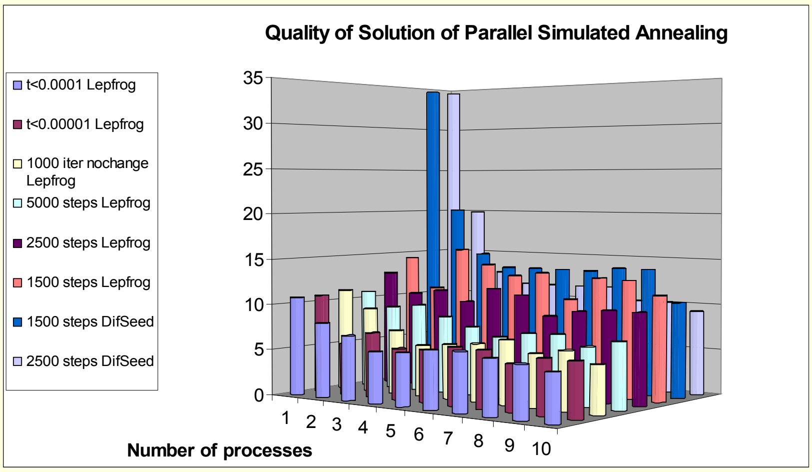
The Influence of the Parallel Random Generator and the Stopping Criteria



144

©BOROVSKA

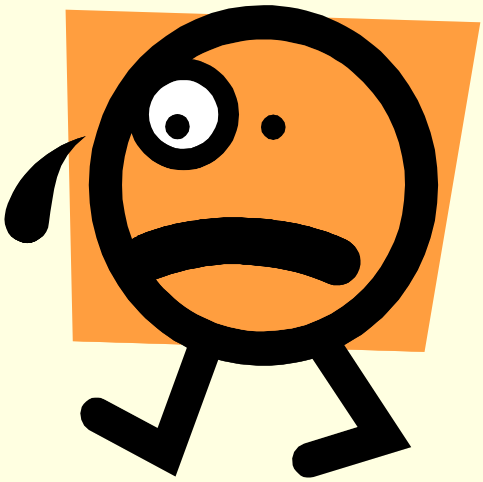
The Influence of the Parallel Random Generator and the Stopping Criteria



Parallel performance

- The speedup gained by parallel independent runs - the results show almost linear speedup when increasing the number of parallel processes.
- The speedup is larger for greater number of persons i.i. greater computational workload in finding the final solution.
- Obviously, utilization of more parallel processes provide better quality solution for fixed number of iterations of the algorithm due to higher diversification of the state space searches provided by the parallel computations in case parallel random generator guarantees uncorrelated or slightly correlated sequences on the different processors.
- The solution quality is better for greater parallel machine size and larger parallel computational workload (the number of iterations), i.e. the parallel algorithm scales well with respect to the size of the parallel machine and the size of parallel application.

Thank You for the Attention!



147