



High Performance Computing Users Group Conference
San Jose, March 19-22, 2000

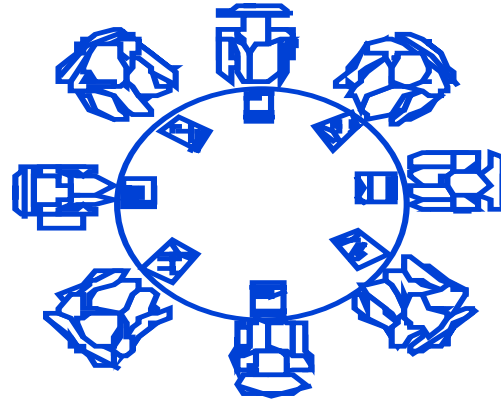
Henry Strauss (GSS FrontOffice Munich)

Comparing Parallel Programming Paradigms and Their Application

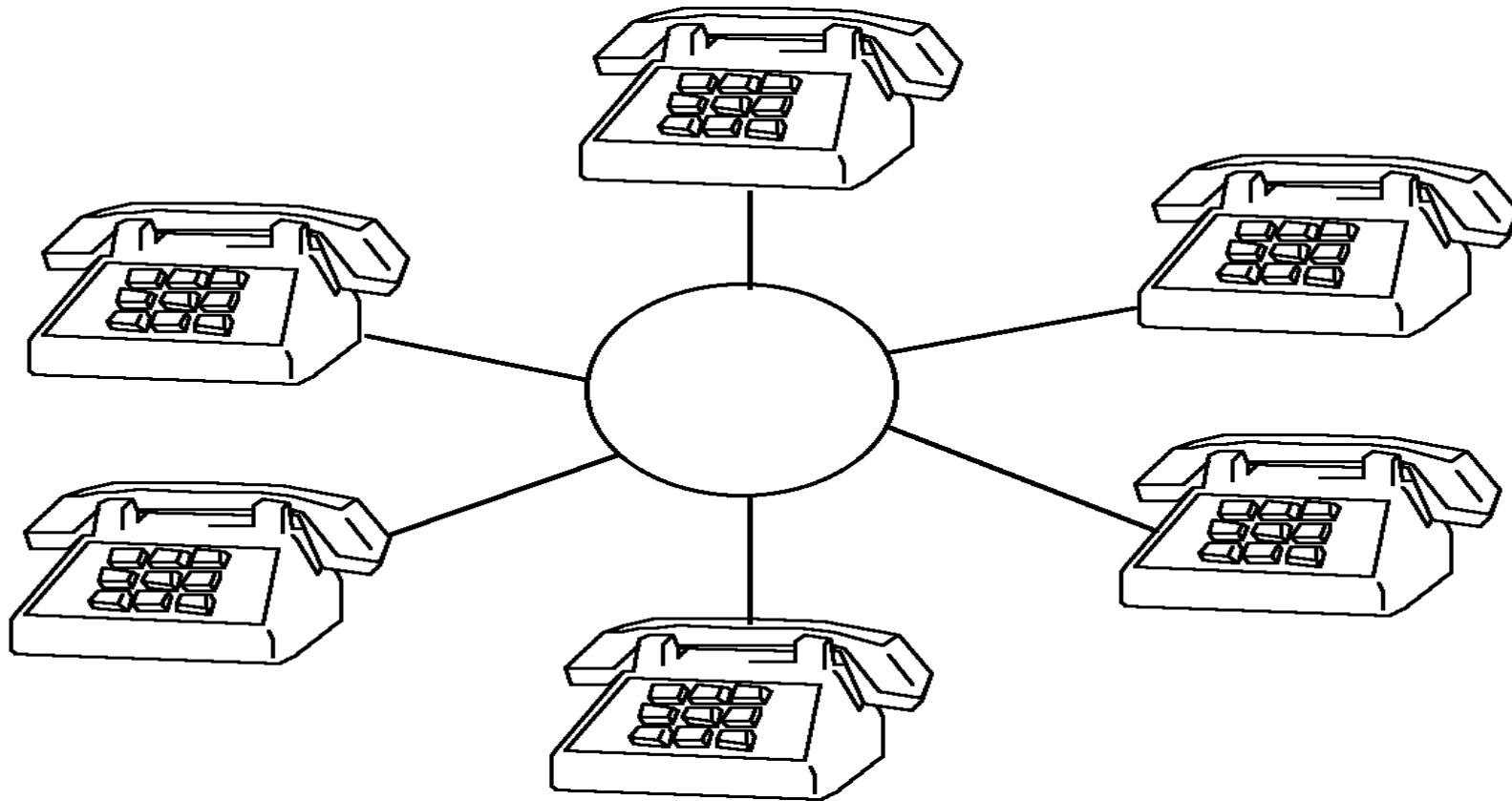
introduction

- many Yins and Yangs:
 - MPP vs SMP
 - local vs global memory machines
 - physically distributed vs virtually shared memory
 - distributed vs shared memory programming
 - SPMD vs MPMD
 - data parallel vs functionally concurrent
 - message passing vs multithreading
 - PVM vs MPI
 - pthreads vs OpenMP
 - static vs dynamic load balancing
 - ...

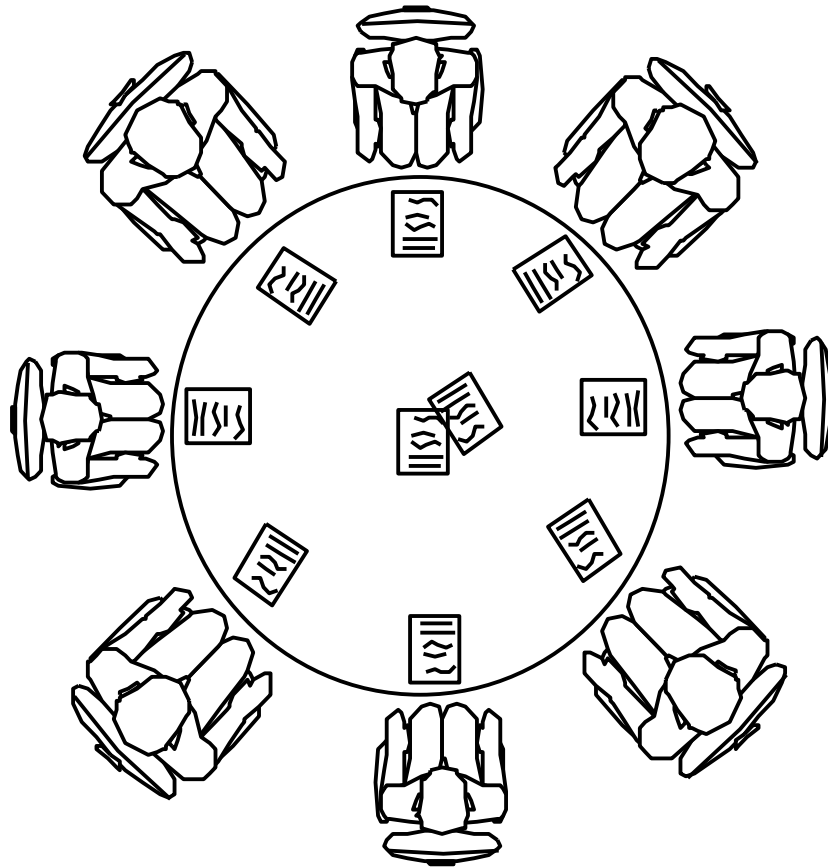
communication



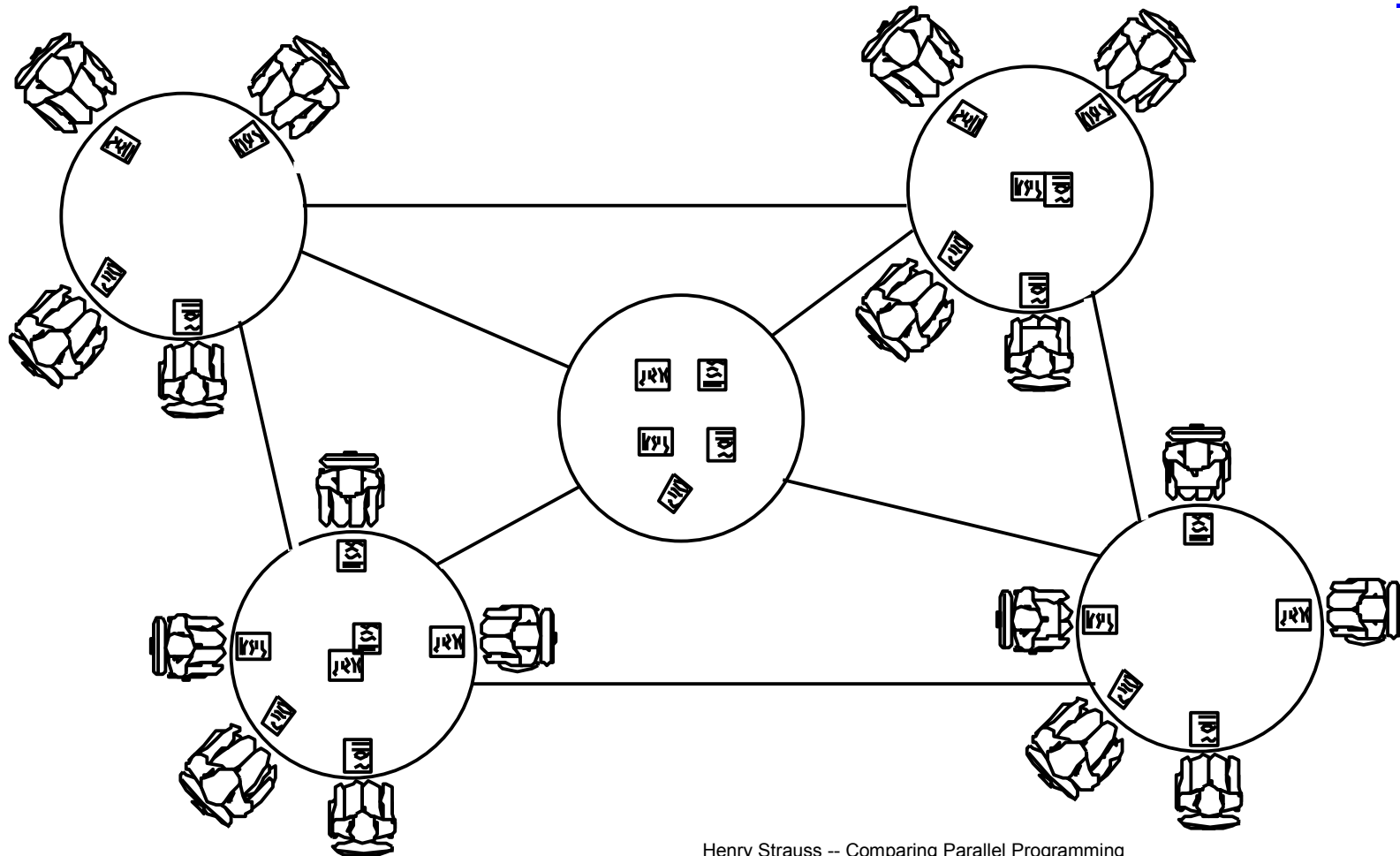
message passing



shared memory



hybrid approach



objectives

- ease-of-use
 - porting existing applications
 - (source code) compatibility
 - support for automatic and explicit methods of ||ism
- compliance with industry standards
 - explicit message passing: (PVM,) MPI
 - shared memory / multithreading: pthreads, OpenMP
- (binary) compatibility with standard Unix o/s
 - availability of applications
 - availability of s/w development tools

PPP I -- automatic compiler-based

- used on SMPs
- fine-grained, i.e. loop-level ||ization only
- app = 1 multithreaded process (=> o/s requirement)
- threads access data via shared memory
- just (re-)compile with appropriate switches
- + ease-of-use
- + no source modification
- (very) limited applicability/efficiency
- (very) low levels of scalability

PPP II -- directive-based

- used on SMPs
- developer gives compiler hints
- app = 1 multithreaded process (=> o/s requirement)
- threads access data via shared memory + thread-private data
- proprietary directives/pragmas or OpenMP
- + compiler can do a better job
- + OpenMP is emerging as a de-facto standard (=> portability)
- more developer effort
- better but generally low levels of scalability

PPP III -- explicit threads-based

- used on SMPs
- placing (pthreads) function calls in source code
- app = 1 multithreaded process (=> o/s requirement)
- threads access data via shared memory + thread-private data
- proprietary threads library or pthreads
- + pthreads = POSIX standard => portability (important for ISVs)
- + total control
- + coarser-grained ||ization possible
- + better efficiency / scalability to higher # of procs
- even more developer effort (prg/alg)

PPP IV -- message passing

- used on SMPs, MPPs, and clusters
- placing (MPI) function calls in source code
- app = multiple processes (possibly multithreaded)
- strictly local memory data
- PVM and other libs replaced by MPI
- + MPI = de-facto standard => portability (important for ISVs)
- + generally good scalability to higher # of procs *if alg fits*
- + can exploit / be implemented on heterogeneous systems
- even more developer effort (alg/prg/dat)

PPP V -- process-based

- used on SMPs, MPPs, and clusters
- placing fork/exec and IPC function calls in source code
- app = multiple processes
- local and global data under user control
- standard Unix functions / system calls
- + portability
- + possibly good scalability
- + can be extended to meta-computing
- more developer effort (alg/prg/dat(+sys))

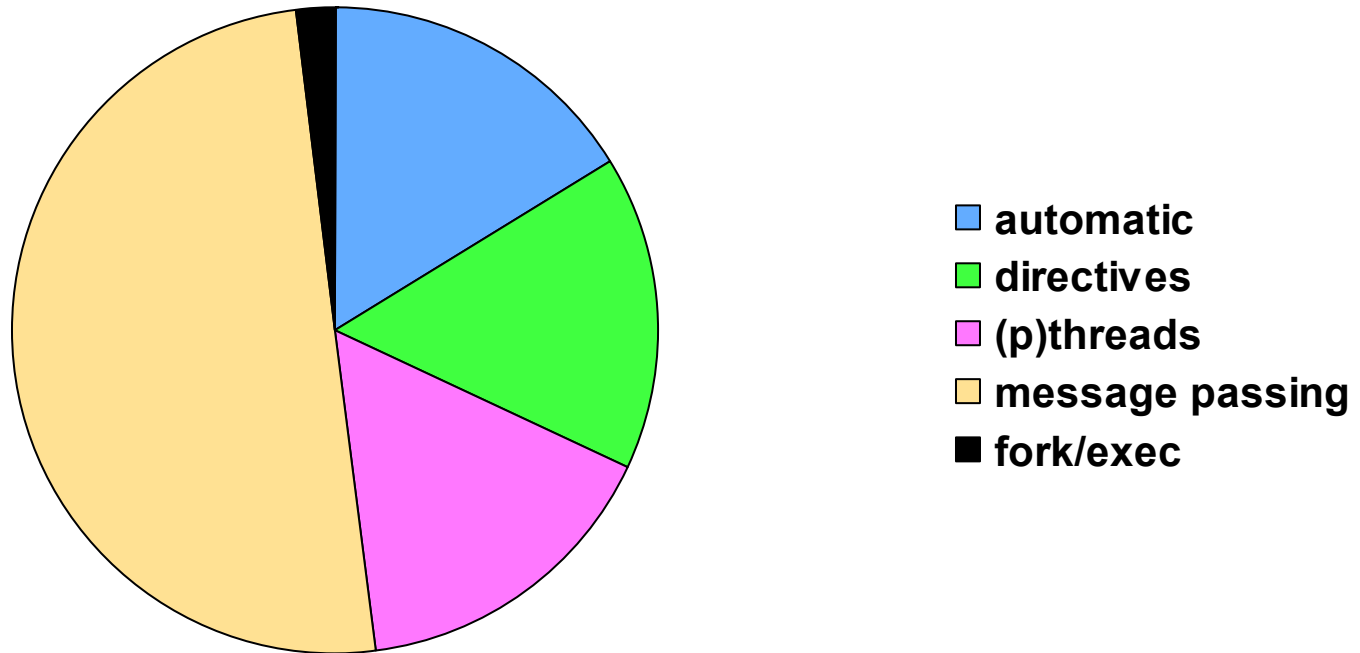
parallel programming standards

-- a comparison from the OpenMP White Paper:

	X3H5	MPI	Pthreads	HPF	OpenMP
scalable	-	+	+/-	+	+
incremental ization	+	-	-	-	+
portable	+	+	+	+	+
Fortran binding	+	+	-	+	+
high level	+	-	-	+	+
supports data ism	+	-	-	+	+
performance oriented	-	+	-	+/-	+

[<http://www.openmp.org/openmp/mp-documents/paper/paper.html>]

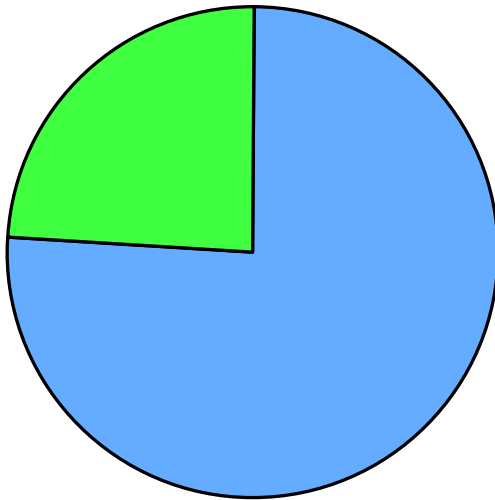
parallel applications by method



source: HPSD Richardson Oct'98

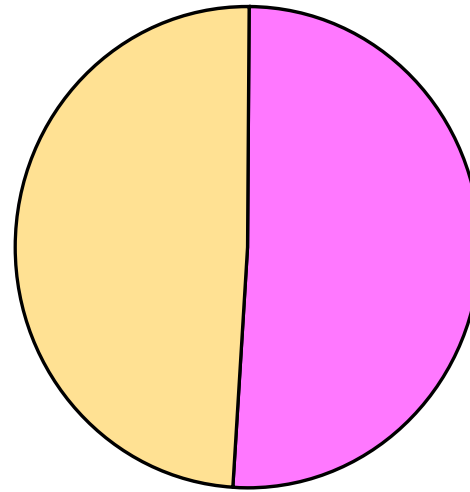
different points of view

explicit vs automatic
parallelization



■ explicit ■ automatic

distributed vs shared memory
programming



■ distributed ■ shared

application field I: CFD

- common problem: simulating a flow
- standard approach for parallelization
computational grid -> domain decomposition -> message passing
=> efficiency satisfactory
static load balancing and sufficiently large problems provided
- advanced software features
multi-block algorithms / dynamic load balancing
local/adaptive mesh refinement
implementable with message passing
BUT cumbersome, errorprone and less efficient

application field II: EDA

- different types of apps
design simulation, layout extraction, design capture, layout capture, design synthesis, layout verification, ...
=> different requirements (cpu, memory, coding)
- currently, ||ization focuses on backend (physical layout)
layout design rule checker (Hercules, Calibre, Dracula, ...)
- these mature apps use message passing or distributed comp.
- new, more general || apps use multithreading

source: Selina Fong, CSD Mountain View

application field III: CC

- quantum chemistry
 - Gaussian: fork/exec, (message passing, threads)
 - Gamess: message passing [tcgmsg -> MPI]
 - Turbomole: message passing [PVM -> MPI], (shared memory)
 - ...
- molecular dynamics & mechanics
 - Charmm: (sockets), message passing [PVM -> MPI]
 - Amber: shared memory, message passing
 - ...

source: Dave Mullally, HPSD Richardson

conclusions

- The PPP of choice is application-dependent.
- Message passing / domain decomposition is the most widely used single PPP, partially for performance / scalability, but also for historical reasons.
- Its applicability for advanced features of state-of-the-art software packages is limited.
- In these cases, it is necessary to use or combine different programming approaches in order to achieve new levels of functionality and performance.

Yin and Yang in Computer Science

**“The most fruitful developments have always
emerged where two different kinds of thinking met.”
-- Heisenberg**

[Communications of the ACM, April 1998, pp.103-111]

Yin and Yang in Computer Science

*"The most fruitful developments have always emerged
where two different kinds of thinking met."*

—Heisenberg

Medical research has produced interesting results with respect to the contrary qualities of the left and right portions of the human brain. Contrary and complementing qualities also play a significant role in various philosophies—best known under the term "Yin and Yang." This duality principle is also evident in computer science methods and pervades all fields of computing, such as user interfaces or parallel processing. Integration of both qualities offers the prospect of solving more complex problems.

Research on the brain has brought to light important facts about how the brain works and—of particular interest for presentation in this article—about the respective strengths and specializations of its two hemispheres (see discussion in [6, 10], or by Nobel Prize winner John C. Eccles in [9]). The left brain, for instance, is responsible for language, rationalism, and differentiation, whereas the right brain supports visualization, emotions, and holism. Table 1 lists the most important differences in orientation between the two hemispheres, mostly determined by tests on people whose brains had been affected by injury or surgery.

Studies of brain signals have confirmed the differences between the hemispheres by finding specific characteristics and basic differences in the alpha band of signals recorded from the left and right brain,

depending on the orientation of the task performed by the test person [6]. This orientation was quite evident for language and emotions: people totally lost their ability to communicate if the left brain was completely cut off, and their ability to make decisions was lost if their right brain's emotional center was destroyed [3]. From this and other neuropsychological tests, however, not only the specializations (often called dichotomies in the literature) became clear, but also the fact that

both hemispheres—and a proper interaction between them—are needed to make up the complete personality of a human being. These insights about the two hemispheres led to a deeper understanding and awareness of the human psyche and intelligence, showing how human beings can develop their full potential. Hubert and Stuart Dreyfus, for example, in their famous book *Mind over Machine* [4] have presented numerous examples of human learning, demonstrating that the rule-based

approach of the left brain keeps a person at the stage of a beginner, whereas to become an expert, it is necessary to integrate situational know-how—a faculty more oriented to the right brain. Sagan has argued that the most significant creative achievements in our culture—legislation and ethical systems, art and music, science and technology—resulted from a cooperation between the left and right brain [10].

The results of modern brain research confirm what ancient Eastern and some modern Western philoso-

