Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Idealized Parallel Computers

Programmierung Paralleler und Verteilter Systeme (PPV)

Sommer 2015

Frank Feinbube, M.Sc., Felix Eberhardt, M.Sc.,
Prof. Dr. Andreas Polze

Theoretical Models for Parallel Computers

Simplified parallel machine model, for theoretical investigation of algorithms

- Difficult in the 70‘s and 80‘s due to large diversity in parallel hardware design

Should improve algorithm robustness by avoiding optimizations to hardware layout specialities (e.g. network topology)

Resulting computation model should be independent from programming model

Vast body of theoretical research results

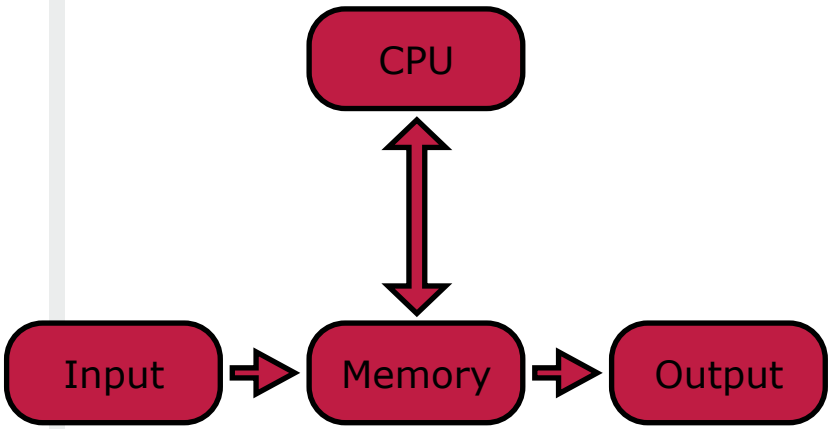
Typically, formal models adopt to hardware developments

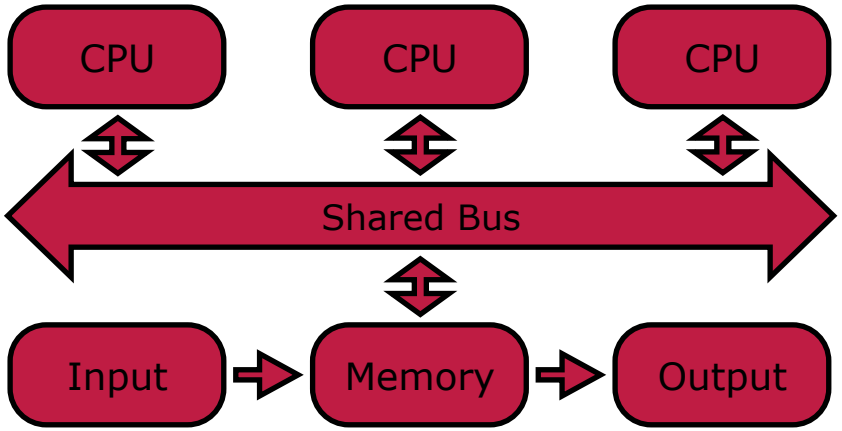
(Parallel) Random Access Machine

RAM assumptions: Constant memory access time, unlimited memory

PRAM assumptions: Non-conflicting shared bus, no assumption on synchronization support, unlimited number of processors

Alternative models: BSP, LogP





PRAM Extensions

Rules for memory interaction to classify hardware support of a PRAM algorithm

Note: Memory access assumed to be in lockstep (synchronous PRAM)

Concurrent Read, Concurrent Write (CRCW)

- Multiple tasks may read from / write to the same location at the same time

Concurrent Read, Exclusive Write (CREW)

- One thread may write to a given memory location at any time

Exclusive Read, Concurrent Write (ERCW)

- One thread may read from a given memory location at any time

Exclusive Read, Exclusive Write (EREW)

- One thread may read from / write to a memory location at any time

PRAM Extensions

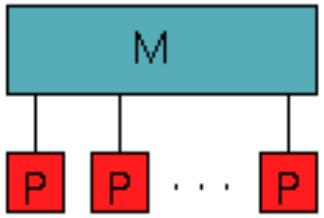
Concurrent write scenario needs further specification by algorithm

- Ensures that the same value is written
- Selection of arbitrary value from parallel write attempts
- Priority of written value derived from processor ID
- Store result of combining operation (e.g. sum) into memory location

PRAM algorithm can act as starting point (unlimited resource assumption)

- Map ‚logical' PRAM processors to restricted number of physical ones
- Design scalable algorithm based on unlimited memory assumption, upper limit on real-world hardware execution
- Focus only on concurrency, synchronization and communication later

PRAM



PRAM extensions

Exclusive-read, exclusive-write (EREW) PRAM

- exklusiver Zugang zu jedem Speicherwort
- schwächstes Modell: Speicherverwaltung muß Minimum an Nebenläufigkeit unterstützen

Concurrent-read, exclusive-write (CREW) PRAM

- mehrere simultane Lesezugriffe auf Speicherwort
- Schreibzugriffe serialisiert

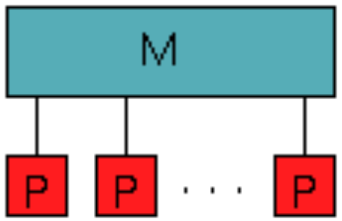
Exclusive-read, concurrent-write (ERCW) PRAM

- parallele Schreibzugriffe auf dasselbe Speicherwort
- Lesezugriffe serialisiert

Concurrent-read, concurrent-write (CRCW) PRAM

- mächtigstes Modell
- kann auf EREW simuliert werden

PRAM



PRAM write operations

Protokolle für parallele Schreibzugriffe

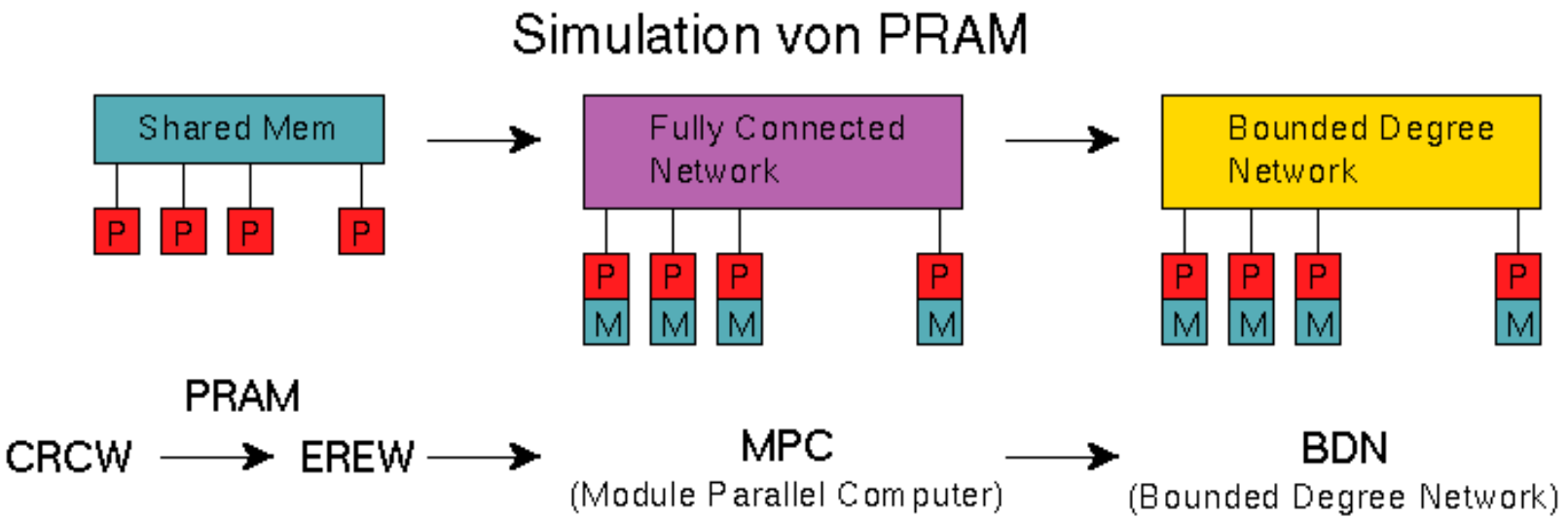
Common: nebenäufiges Schreiben ist erlaubt, wenn alle Prozessoren denselben Wert schreiben

Arbitrary: einer der Prozessoren kann erfolgreich schreiben, die anderen scheitern

Priority: Prozessoren haben vorherdefinierte Priorität, Prozessor mit höchstem Wert hat Erfolg, die anderen scheitern

Sum: Die Summe aller Werte wird geschrieben
(Combining: jeder assoziative Operator ist denkbar)

PRAM Simulation



			Deterministische Simulation	randomized Simulation
CRCW	→	EREW	O(log n)	
EREW	→	MPC	O(log n/log log n)	O(log log n log*n)
CRCW	→	MPC	O(log n)	O(log n)
MPC	→	BDN	$O(\log^2 n)$	O(log n)
EREW	→	BDN	$O(\log^2 n/\log \log n)$	O(log n)
CRCW	→	BDN	$O(\log^2 n/\log \log n)$	O(log n)

Tim J. Harris, *A Survey of PRAM Simulation Techniques*, ACM Comp. Surveys, Vol. 26, No. 2, June 1994

Example: Parallel Sum

General parallel sum operation works with any associative and commutative combining operation (multiplication, maximum, minimum, logical operations, ...)

- Typical reduction pattern

PRAM solution: Build binary tree, with input data items as leaf nodes

- Internal nodes hold the sum, root node as global sum
- Additions on one level are independent from each other
- PRAM algorithm: One processor per leaf node, in-place summation
- Computation in $O(\log_2 n)$

```
int sum=0;
for (int i=0; i<N; i++) {
   sum += A[i];
}
```

Example: Parallel Sum

```
for all l levels (1..log2n){
  for all i items (0..n-1) {
    if (((i+1) mod 2^l) = 0) then
      X[i] := X[i-2^(l-1)]+X[i]
  }
}
```

Example: n=8:

- l=1: Partial sums in X[1], X[3], X[5], [7]
- l=2: Partial sums in X[3] and X[7]
- l=3: Parallel sum result in X[7]

Correctness relies on PRAM lockstep assumption (no synchronization)

Bulk-Synchronous Parallel (BSP) Model

Leslie G. Valiant. A Bridging Model for Parallel Computation, 1990

Success of von Neumann model

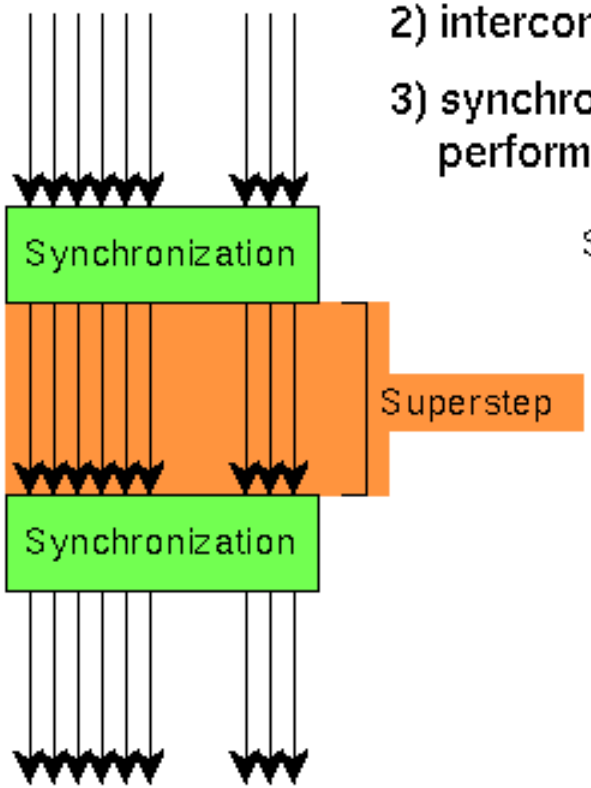
- Bridge between hardware and software
- High-level languages can be efficiently compiled based on this model
- Hardware designers can optimize the realization of this model

Similar model for parallel machines

- Should be neutral about the number of processors
- Program are written for *v* virtual processors that are mapped to *p* physical ones, were *v* >> *p* -> chance for the compiler

Hasso Plattner Institut

Bulk Synchronous Parallel Model



1) processor/memory modules

2) interconnection network

3) synchronizer,
   performs barrier synchronization

Superstep: - Prozessor vollführt lokale Berechnungen
- empfängt/sendet Nachrichten

jeder Prozessor kann höchstens h-Nachrichten empfangen und h-Nachrichten senden
(h - relation)

L.G.Valiant, *A Bridging Model for Parallel Computation*, CACM, 33(8), 103-11, August 1990

Bulk-Synchronous Parallel (BSP) Model

Bulk-synchronous parallel computer (BSPC) is defined by:

- *Components*, each performing processing and / or memory functions
- *Router* that delivers messages between pairs of components
- Facilities to synchronize components at regular intervals *L (periodicity)*

Computation consists of a number of *supersteps*

- Each L, global check is made if the *superstep* is completed

Router concept splits computation vs. communication aspects, and models memory / storage access explicitely

Synchronization may only happen for some components, so long-running serial tasks are not slowed down from model perspective

L is controlled by the application, even at run-time

LogP

Culler et al., LogP: Towards a Realistic Model of Parallel Computation, 1993

Criticism on overly simplification in PRAM-based approaches, encourage exploitation of ‚formal loopholes' (e.g. no communication penalty)

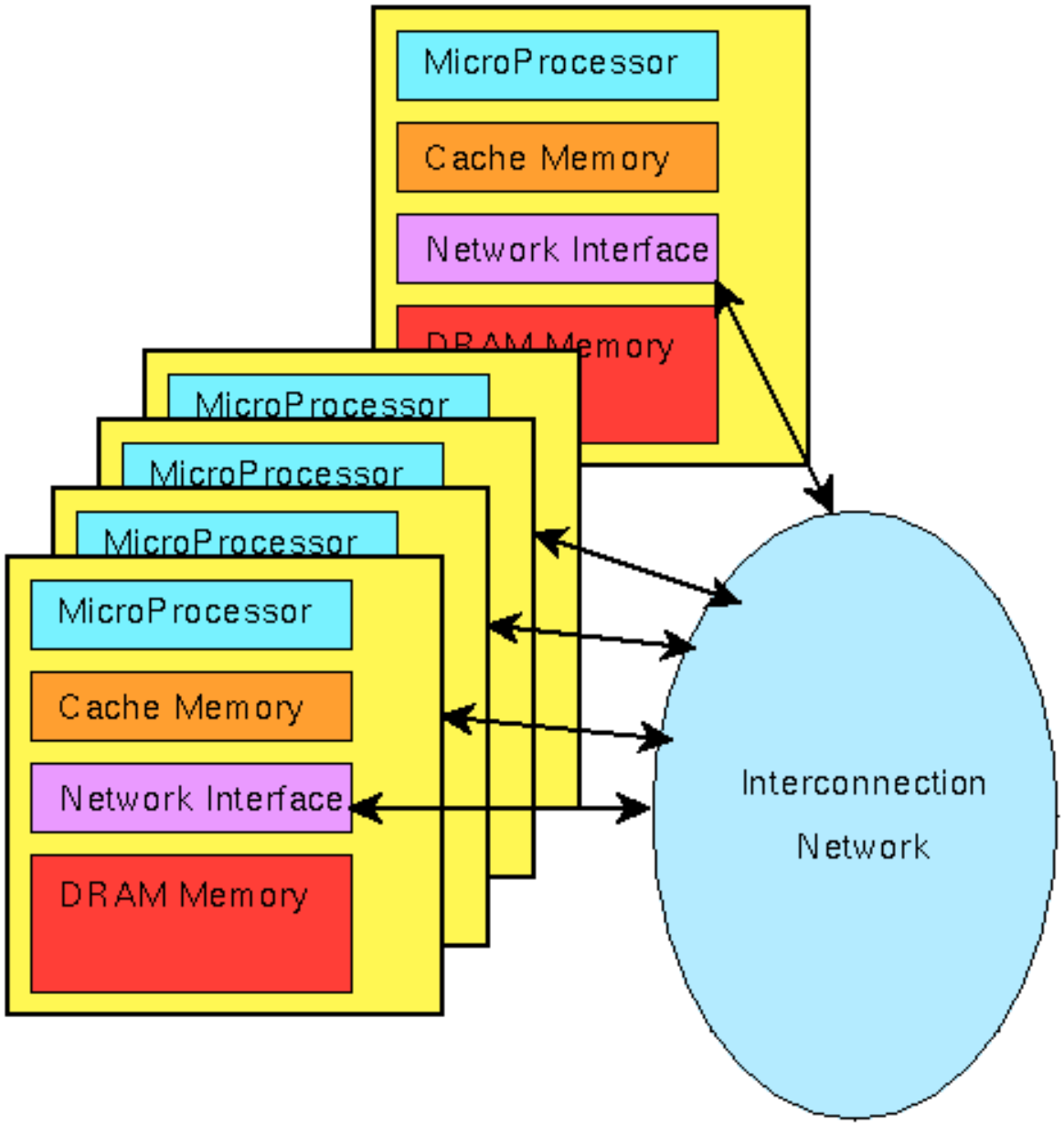
Trend towards multicomputer systems with large local memories

Characterization of a parallel machine by:

- **P**: Number of processors
- **g**: Gap: Minimum time between two consecutive transmissions
  - Reciprocal corresponds to per-processor communication bandwidth
- **L**: Latency: Upper bound on messaging time from source to target
- **o**: Overhead: Exclusive processor time needed for send / receive operation

L, o, G in multiples of processor cycles

LogP architecture model



Architectures that map well on LogP:

- Intel iPSC, Delta, Paragon,
- Thinking Machines CM-5, Ncube,
- Cray T3D,
- Transputer MPPs: MeikoComputing Surface, Parsytec GC.

LogP

Analyzing an algorithm - must produce correct results under all message interleaving, prove space and time demands of processors

Simplifications

- With infrequent communication, bandwidth limits (g) are not relevant
- With streaming communication, latency (L) may be disregarded

Convenient approximation: Increase overhead (o) to be as large as gap (g)

Encourages careful scheduling of computation, and overlapping of computation and communication

Can be mapped to shared-memory architectures

- Reading a remote location requires *2L+4o* processor cycles

LogP

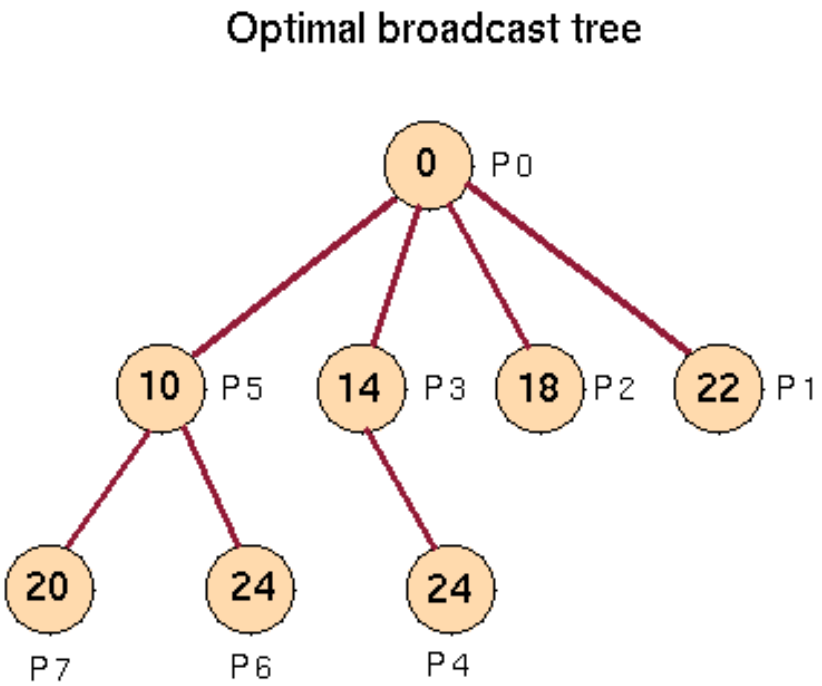
Matching the model to real machines

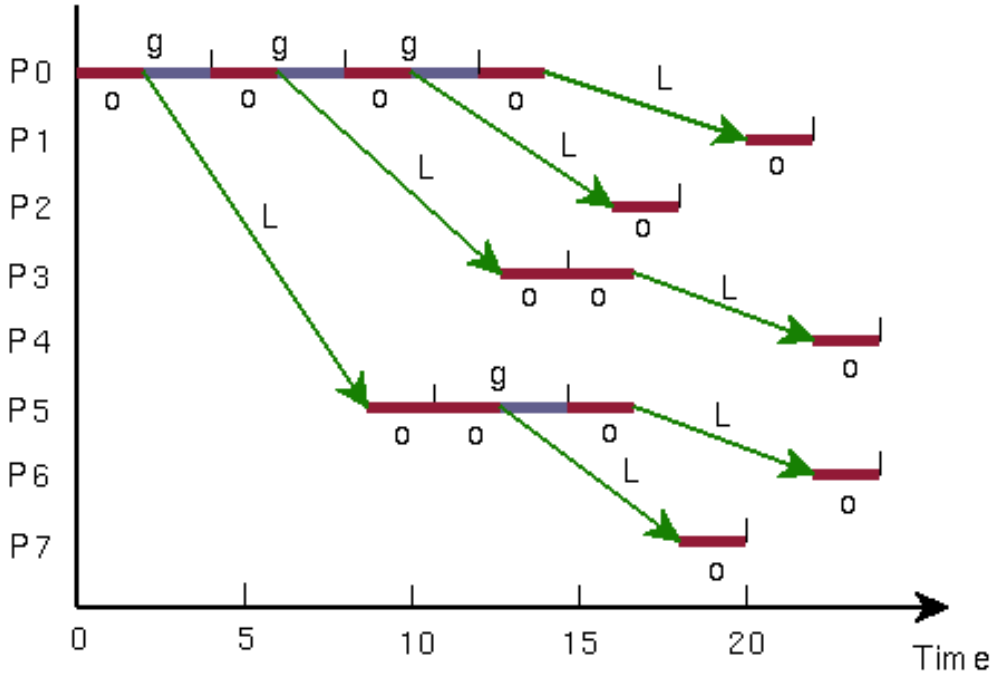
- Saturation effects: Latency increases as function of the network load, sharp increase at saturation point - captured by capacity constraint
- Internal network structure is abstracted, so ‚good' vs. ‚bad' communication patterns are not distinguished - can be modeled by multiple *g's*
- LogP does not model specialized hardware communication primitives, all mapped to send / receive operations
- Separate network processors can be explicitly modeled

Model defines 4-dimensional parameter space of possible machines

- Vendor product line can be identified by a curve in this space

LogP – optimal broadcast tree





LogP – optimal summation