INFORMATIK

8 Lastverteilung

- Einführung
- Zufälliges Zuordnen
- Zentralisierte dynamische Lastverteilung
- Work Stealing
- Nächster-Nachbar-Austausch

Lastverteilung was ist das?

Gleichmäßige Verteilung unregelmäßiger Berechnungen auf die Prozessoren.

Da hören die Gemeinsamkeiten schon auf:

- Wie teuer ist die Übertragung von Teilberechnungen?
- Wie gut kann man Rechenaufwand der Teile abschätzen?
- Ändert sich die Lastsituation dynamisch?
- Kann man Probleme beliebig unterteilen?
- Sind die Teile unabhängig (Größe, Kommunikation)?
- Kann man Lastverteilungsentscheidungen rückgängig machen?

$\Rightarrow$ Es gibt generische Techniken aber keine erschlägt alle Anwendungen.

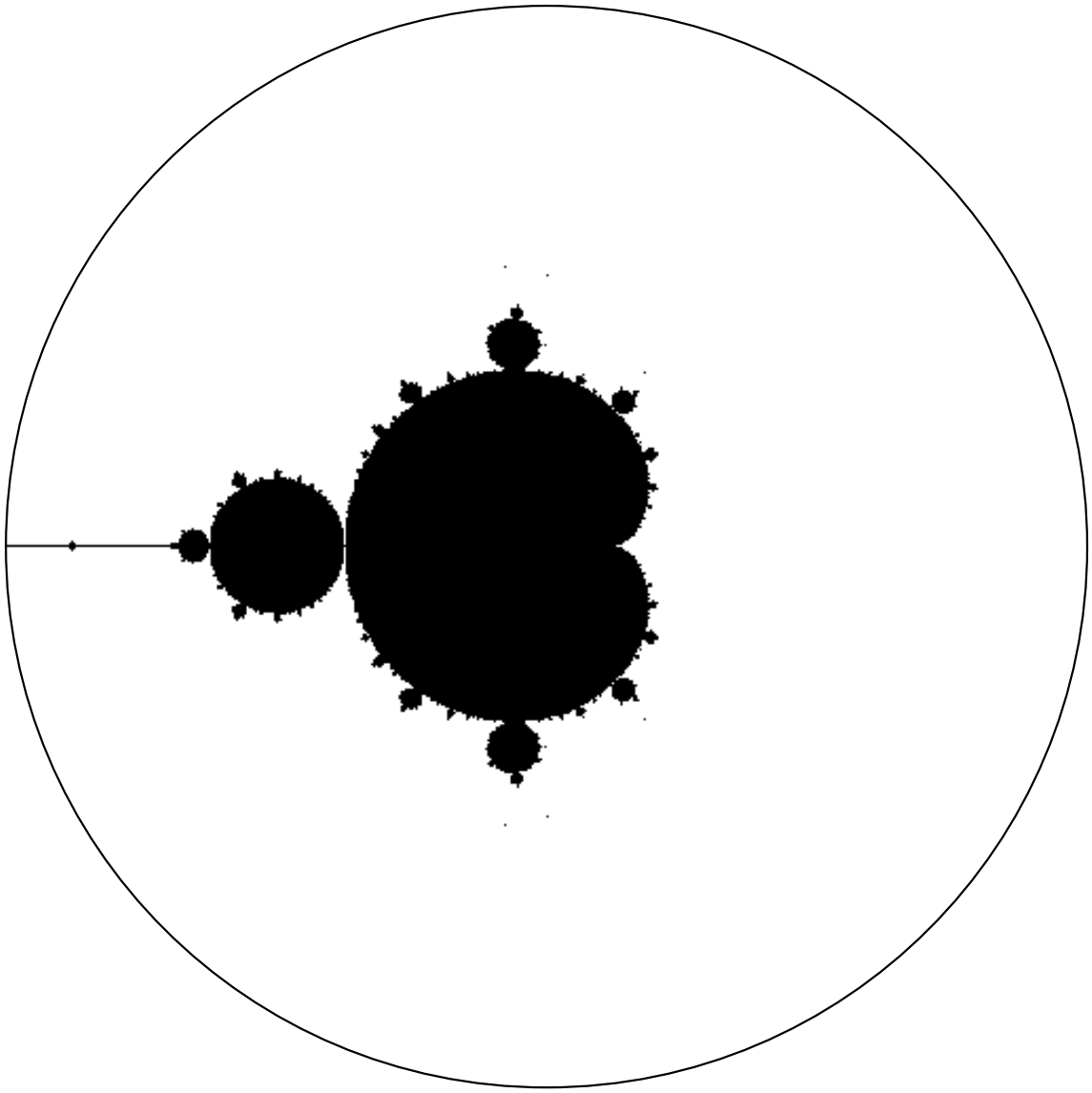
Gute Maschinen- und Problemmodelle sind gefragt

8.1 Beispiel Mandelbrotmenge

$$z_c(m) : \mathbb{N} \to \mathbb{C}$$

$$z_c(0) := 0, \quad z_c(m+1) := z_c(m)^2 + c$$

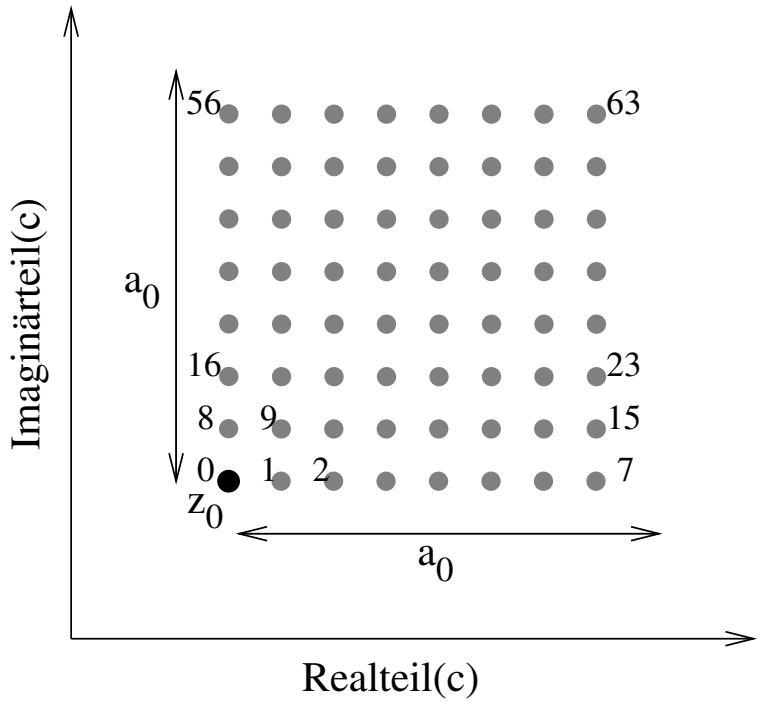
$$M := \{c \in \mathbb{C} : z_c(m) \text{ ist beschränkt}\} .$$



INFORMATIK

8.1.1 Angenäherte Berechnung

- Berechung nur für quadratischen Ausschnitt der komplexen Zahlenebene
- Berechnung nur für diskretes Gitter von Punkten
- z_c unbeschränkt falls $|z_c(k)| \geq 2$
- Abbruch nach $m_{\max}$ Iterationen



Wo liegt das Lastverteilungsproblem?

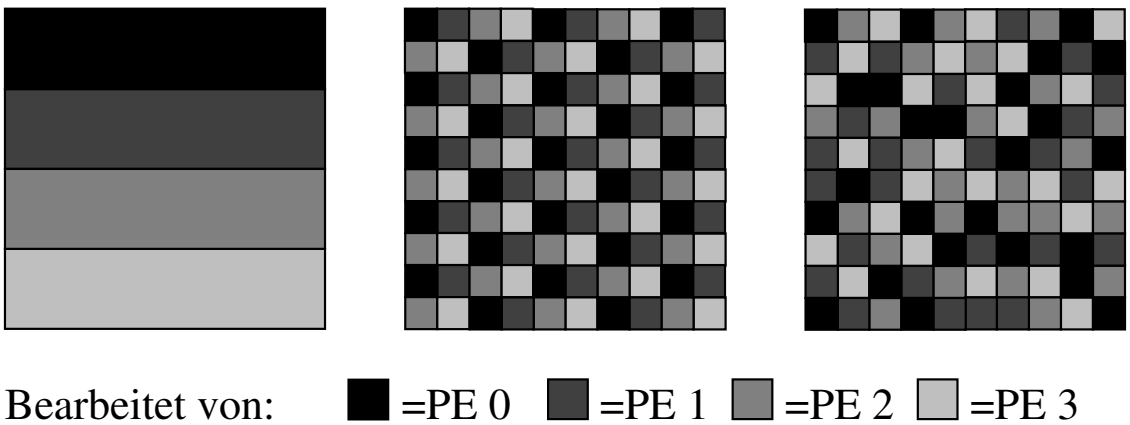
8.1.2 Code

```
int iterate(int pos, int resolution, double step)
{
  int iter;
  complex c =
    z0 + complex((double)(pos % resolution)
                 * step,
                 (double)(pos / resolution)
                 * step);
  complex z = c;
  for (iter = 1;
       iter < maxiter && abs(z) <= LARGE;
       iter++)
  {
    z = z*z + c;
  }
  return iter;
}
```

8.1.3 Statische Äpfelverteilung

Da kaum Kommunikation stattfindet sind wir sehr flexibel

- Streifenweise Zerlegung
  - Warum attraktiv?
  - Warum besser nicht?
- zyklisch. Gut aber können wir das beweisen?
- Zufällig



8.1.4 Parallelisierung der Zuordnungsphase

- Wenn die Teilprobleme irgendwie auf die PEs verteilt sind: Zufallspermutation via all-to-all. (Siehe auch sample sort)
- Implizites Erzeugen der Einzelteile
  - Teilproblem läßt sich allein aus seiner Nummer $1 \ldots n$ erzeugen.
  - Problem: Parallele Berechnung einer (Pseudo)Zufallspermutation

Zufallsgeneratoren auf der Basis von linearen Kongruenzen lassen sich “vorspulen”.

8.2 Zufälliges Zuordnen

- Gegeben: n Teilprobleme der Größe $l_1, \ldots, l_n$
- Sei $L := \sum_{i \leq n} l_i$
- Sei $l_{\max} := \max_{i \leq n} l_i$
- Ordne die Teilprobleme zufälligen PEs zu

Satz: Falls

$$L \geq 2P(\beta + 1)l_{\max}(\ln P)/(\epsilon^2) + O(\epsilon^3))$$

dann ist die maximale Last höchstens

$$(1 + \epsilon)L/P$$

mit Wahrscheinlichkeit mindestens $1 - P^{-\beta}$.
Beweis: ... Chernoff-Schranken ...

Diskussion

+ Teilproblemgrößen müssen nicht bekannt sein
+ Es ist unerheblich wo die Teilprobleme herkommen (verteilte Erzeugung möglich)
– Unakzeptable bei großem $l_{\max}$
– Sehr gute Lastverteilung nur bei sehr großem $L/l_{\max}$ (quadratisch in $1/\epsilon$.

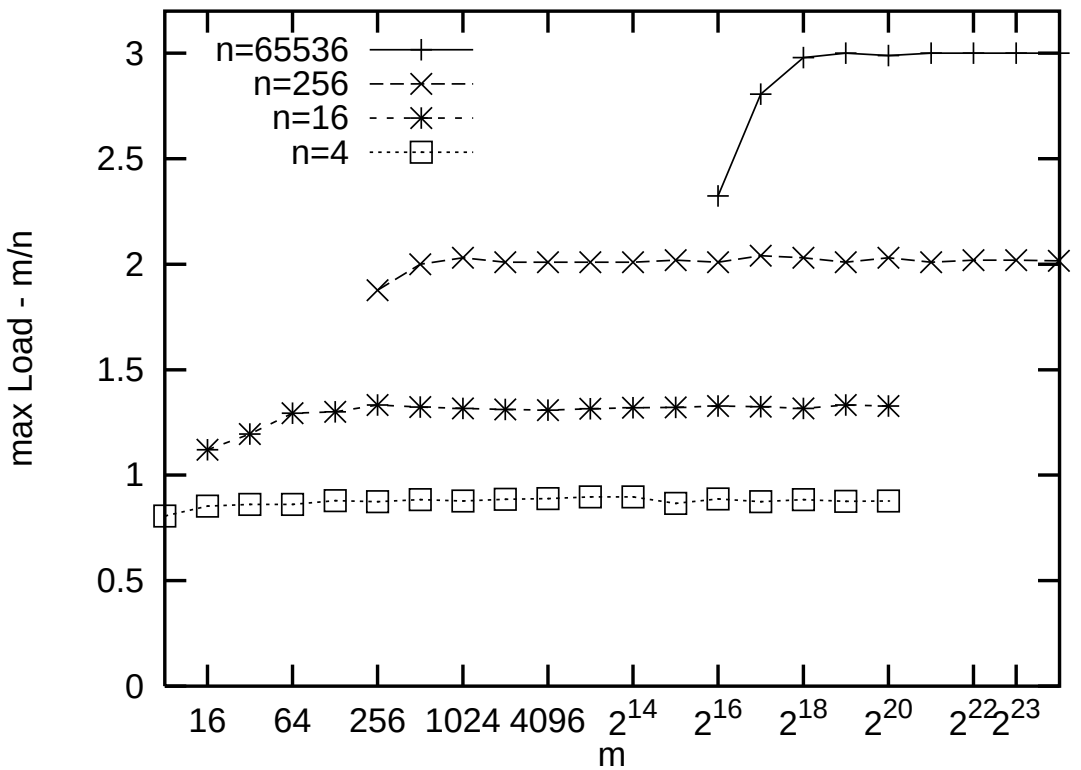
Anwendungsbeispiel: Airline Crew Scheduling

Eine einzige zufällige Verteilung löst k simultane Lastverteilungsprobleme. (Deterministisch vermutlich ein schwieriges Problem.)

INFORMATIK

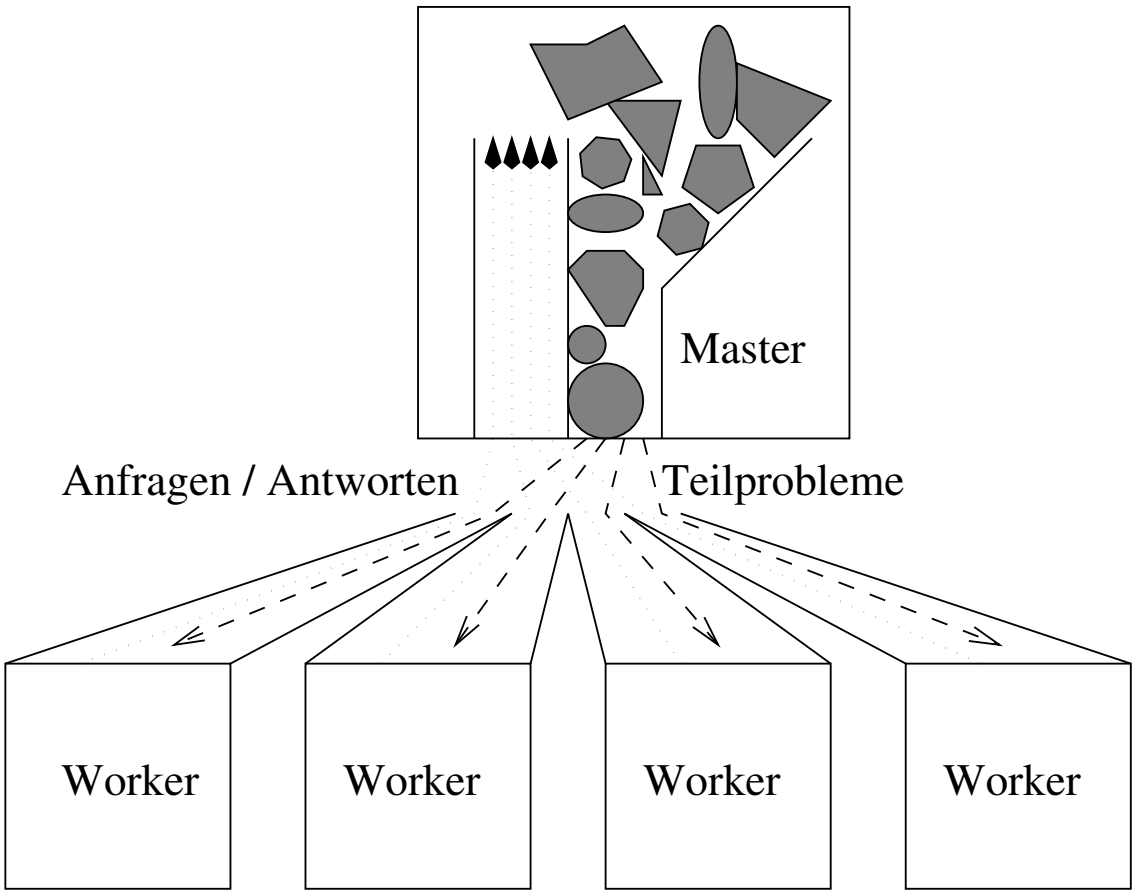
8.2.1 Zufällige balancierte Zuordnung

- "Jobs" kommen nacheinander und sollen sofort zugeordnet werden
- Det. Verfahren: suche minimal belastetes PE — zu teuer
- Zufällig nicht sehr gut.
- Wähle zufällig **zwei** potentielle Ziele und alloziere auf dem niedriger belasteten
- Vermutung: $l_{\max} = n/p + \mathcal{O}(\log\log p)$



INFORMATIK

8.3 Das Master-Worker-Schema



- Beliebig unterteilbare Last
- Anfangs auf Master-PE
- Teilproblemgrößen sind abschätzbar aber nicht genau bekannt
- Einmal abgegebenes Stück kann nicht weiter unterteilt werden (nonpreemptive)

8.3.1 Diskussion

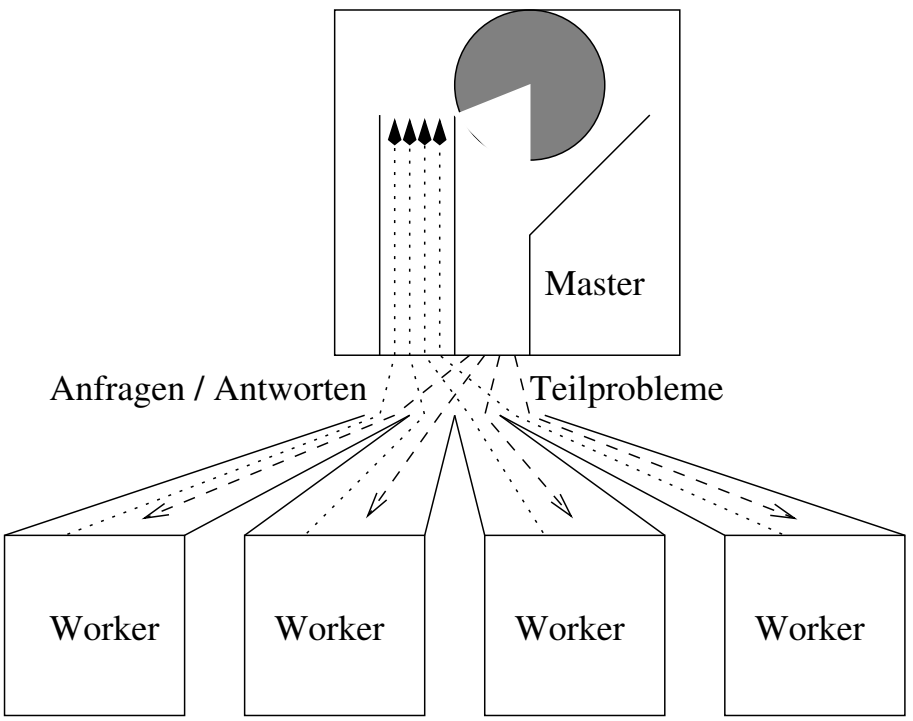
+ Einfach
+ Natürliches Ein- Ausgabeschema (aber u.U. gesonderter Plattensklave)
+ Leicht zu debuggen
– Kommunikationsengpaß ⇒ Tradeoff Kommunikationsaufwand versus Imbalance
– Wie soll aufgespalten werden?
– Multilevelschemata sind kompliziert und nur begrenzt hilfreich

8.3.2 Größe der Teilprobleme

- Möglichst grosse Probleme abgeben solange Lastverteilung nicht gefährdet. Warum?
- Konservatives Kriterium: Eine *obere* Schranke für die Größe des abgegebenen Teilproblems sollte $1/P$-tel einer *unteren* Schranke für die sonst im System befindliche Last nicht übersteigen.

– Woher Grössenabschätzung nehmen

- Aggressivere Verfahren ggf. sinnvoll

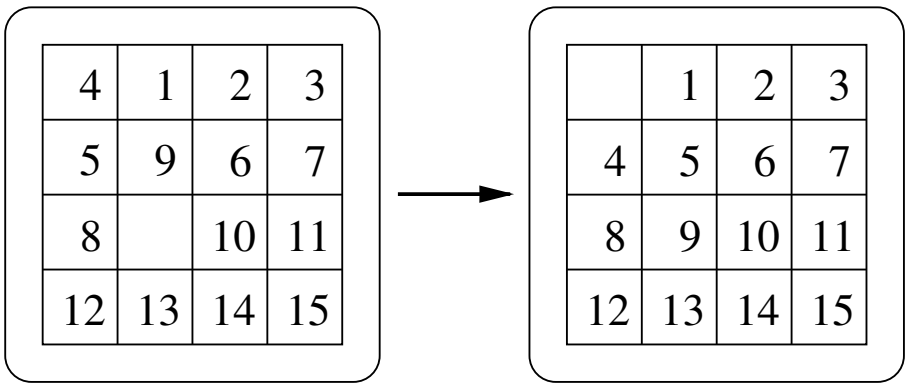


8.4 Work Stealing

- (Fast) Beliebig unterteilbare Last
- Anfangs alles auf PE 0
- Fast nichts bekannt über Teilproblemgrößen
- Preemption erlaubt. (Sukzessives aufspalten)

INFORMATIK

Example: The 15-Puzzle



Korf 85: Iterative deepening depth first search with $\approx 10^9$ tree nodes.

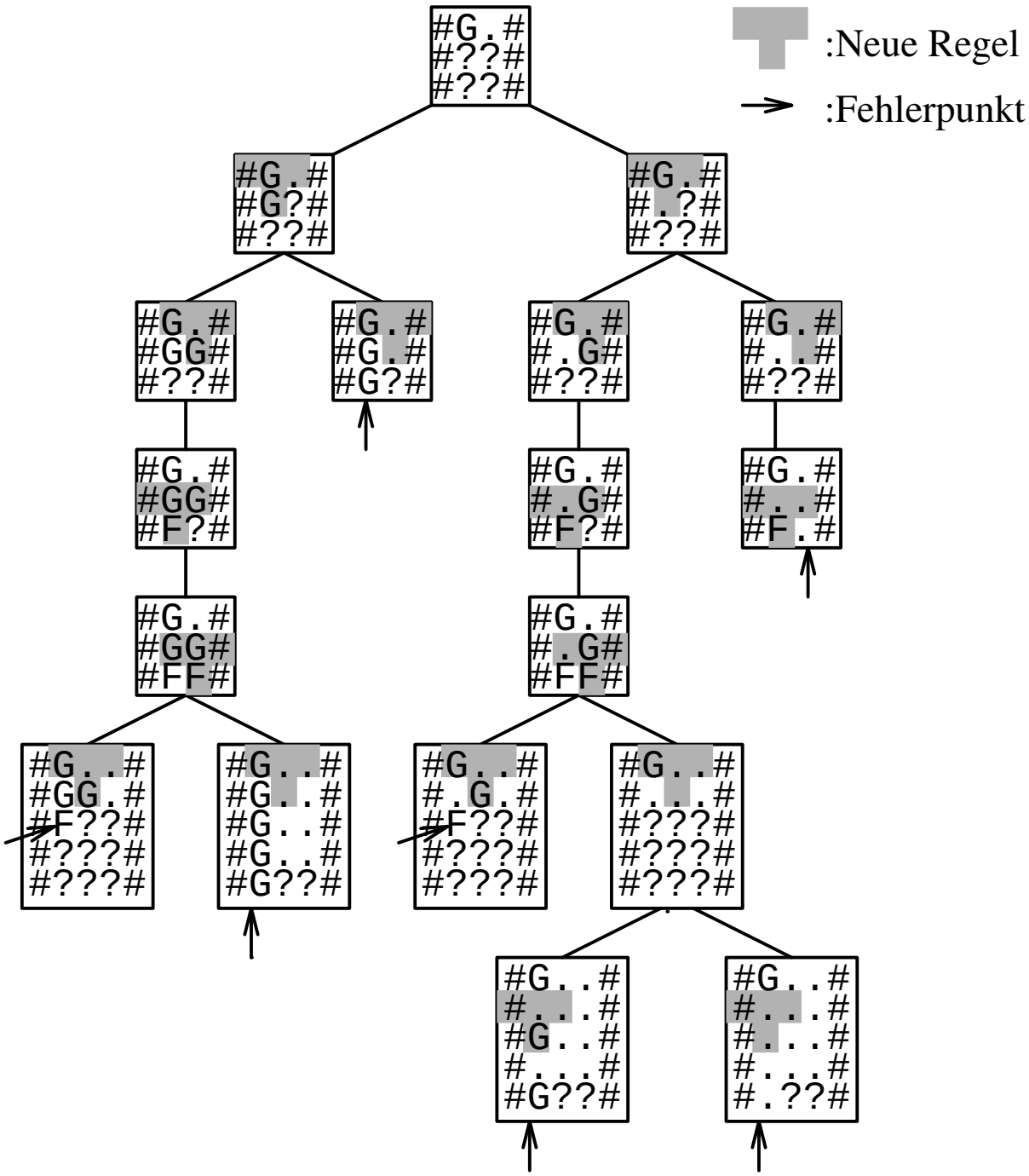
INFORMATIK

Example: The Firing Squad Synchronization Problem

```
#G.#G..#G...#G....#G.....#G......#G.......#
#GG#GX.#GX..#GX...#GX....#GX.....#GX......#
#FF#XXX#XXG.#XXG..#XXG...#XXG....#XXG.....#
   #GGG#GX.G#GX.X.#GX.X..#GX.X...#GX.X....#
   #FFF#XXXX#XXG.X#XXG.G.#XXG.G..#XXG.G...#
       #GGGG#GX.GX#GX.XXG#GX.XXX.#GX.XXX..#
       #FFFF#XXXXX#XXG.XX#XXG.G.X#XXG.G.G.#
            #GGGGG#GX.G.G#GX.XXGX#GX.XXXXG#
            #FFFFF#XXXXXX#XXG.XGX#XXG.GGXX#
                 #GGGGGG#GX.G.GX#GX.XXGXG#
                 #FFFFFF#XXXXXXX#XXG.XGXX#
                       #GGGGGGG#GX.G.GXG#
                       #FFFFFFF#XXXXXXXX#
                             #GGGGGGGG#
                             #FFFFFFFF#
```

INFORMATIK

Backtracking over Transition Functions

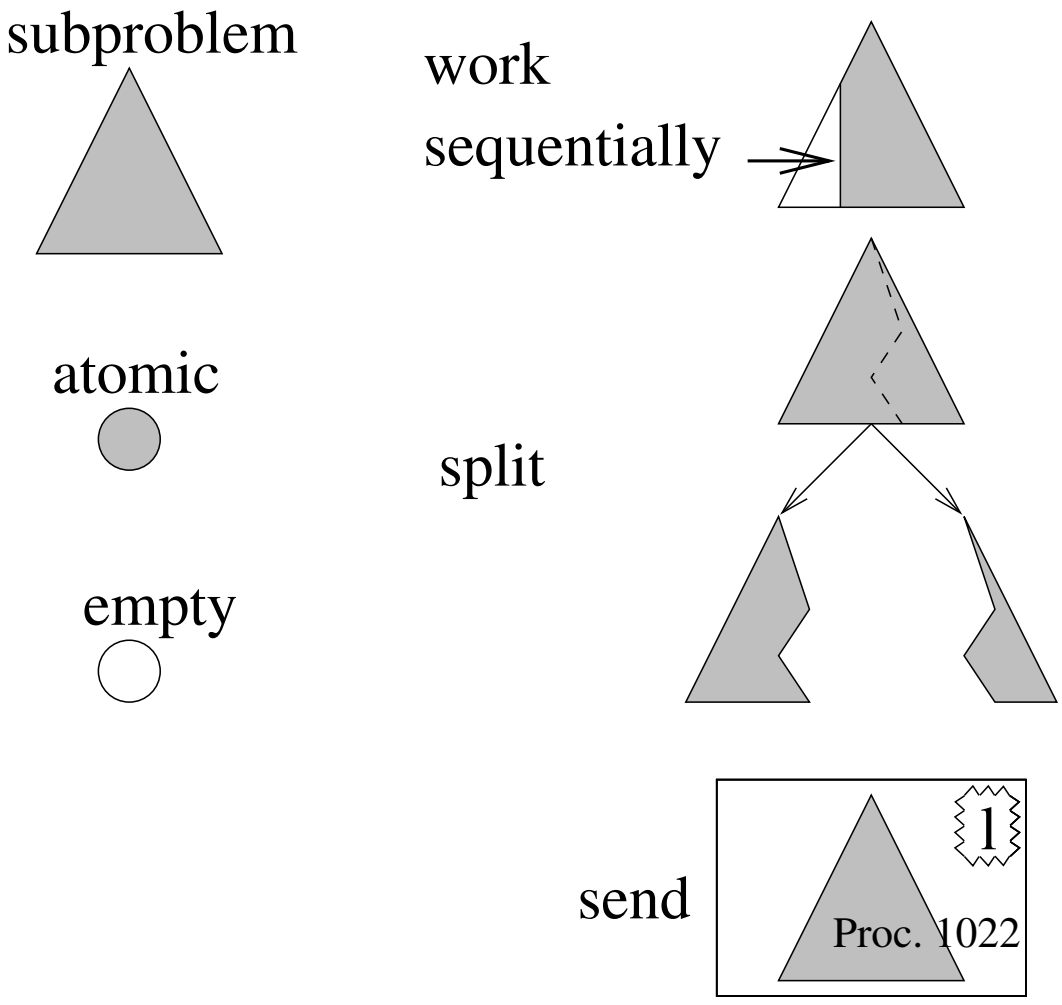


Goal for the analysis

$$T_{\text{par}} \leq (1+\epsilon)\frac{T_{\text{seq}}}{n} + \text{lower order terms}$$

INFORMATIK

An Abstract Model: Tree Shaped Computations

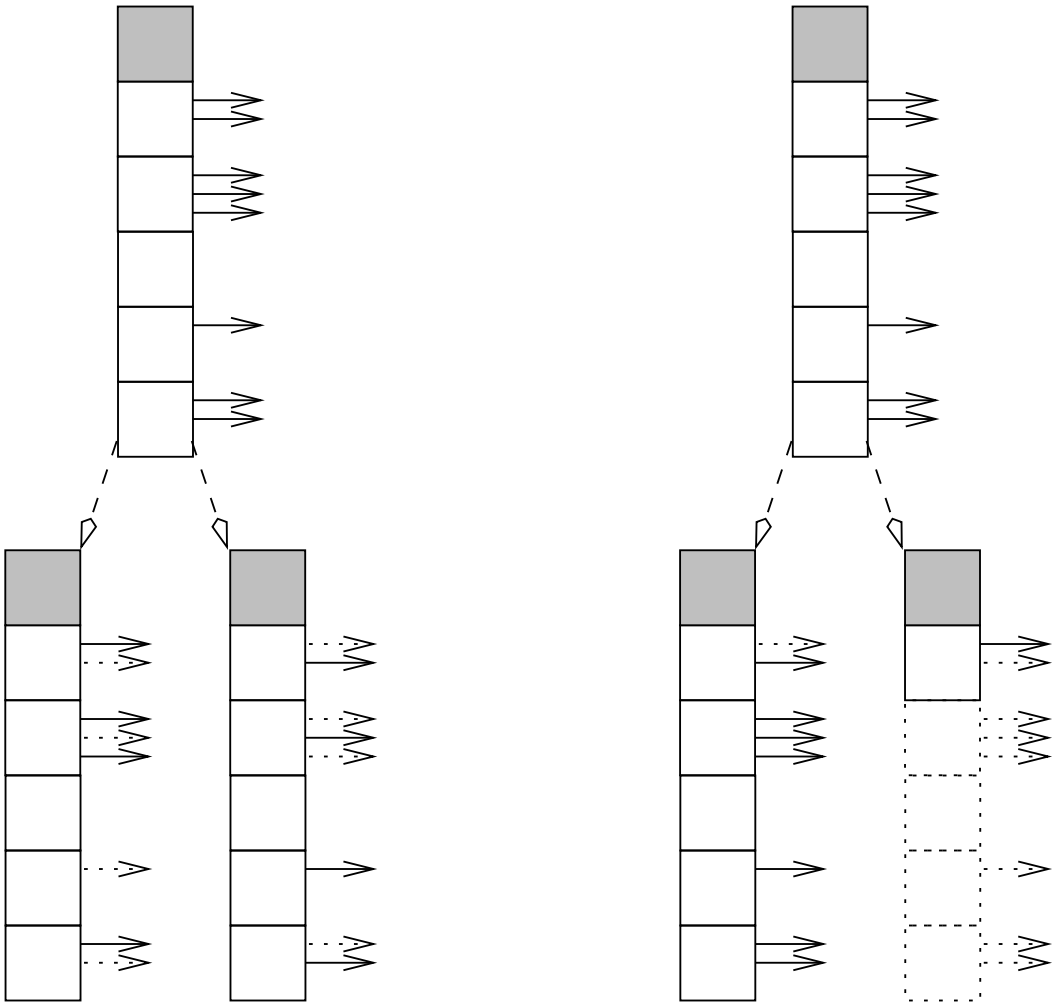


INFORMATIK

Relation to Depth First Search

let stack consist of root node only
while stack is not empty **do**
    remove a node N from the stack
    if N is a leaf **then**
        evaluate leaf N
    else
        put successors of N on the stack
    fi

Splitting Stacks



a) b)

Other Problems Categories

- Loop Scheduling
- Higher Dimensional Interval Subdivision
- Particle Physics Simulation
- Generalization: Multithreaded computations. $h \rightsquigarrow T_\infty$

An Application List

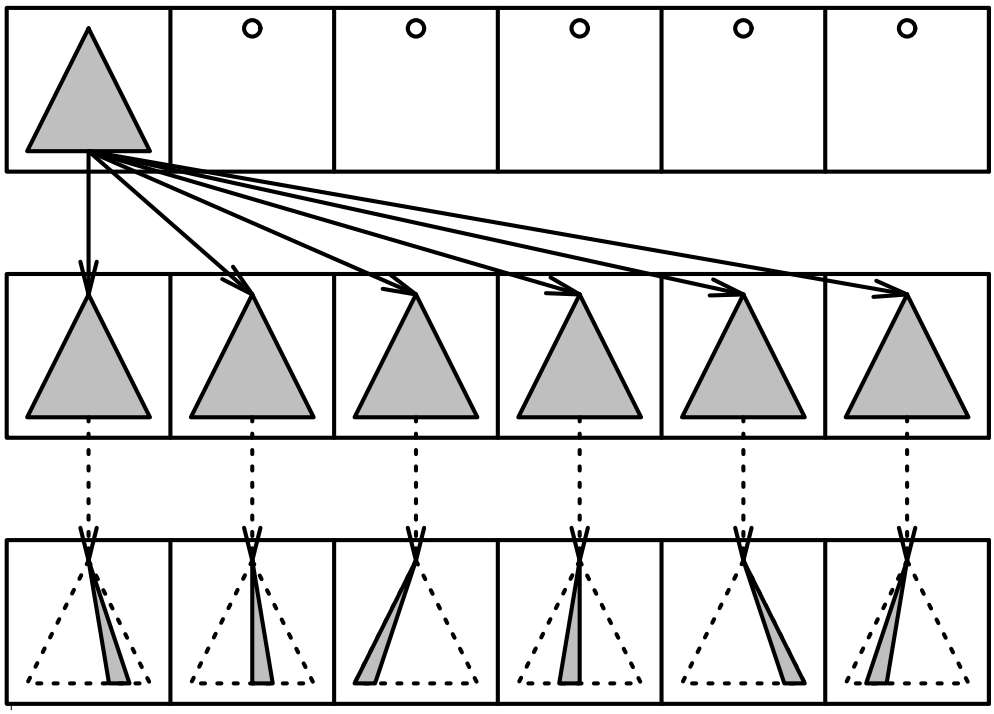
- Discrete Mathematics (Toys?):
  - Golomb Rulers (more later)
  - Cellular Automata, Trellis Automata
  - 15-Puzzle, n-Queens, Pentominoes ...
- In Principle any NP-complete Problem (nondeterminism $\rightsquigarrow$ branching)
  - 0/1 Knapsack Problem (fast!)
  - Quadratic Assignment Problem
  - SAT
- Functional, Logical Programming Languages
- Constraint Satisfaction, Planning, ...
- Numerical: Adaptive Integration, Nonlinear Optimization by Interval Arithmetics, Eigenvalues of Tridiagonal Matrices

Limits of the Model

- Quicksort and similar divide-and-conquer algorithms (shared memory OK $\rightsquigarrow$ Cilk)
- Finding the first Solution (often OK)
- Branch-and-bound
  - Verifying bounds OK
  - Depth-first often OK
- Subtree dependent pruning
  - FSSP OK
  - Game tree search tough (load balancing OK)

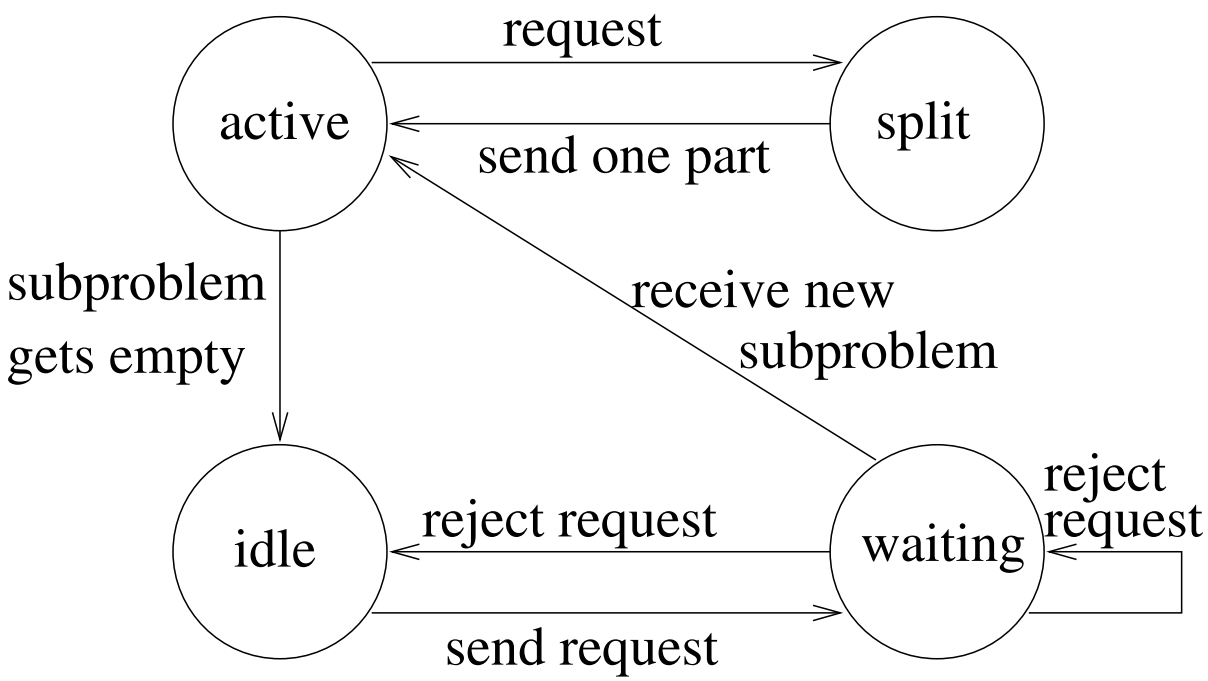
INFORMATIK

Direct Splitting



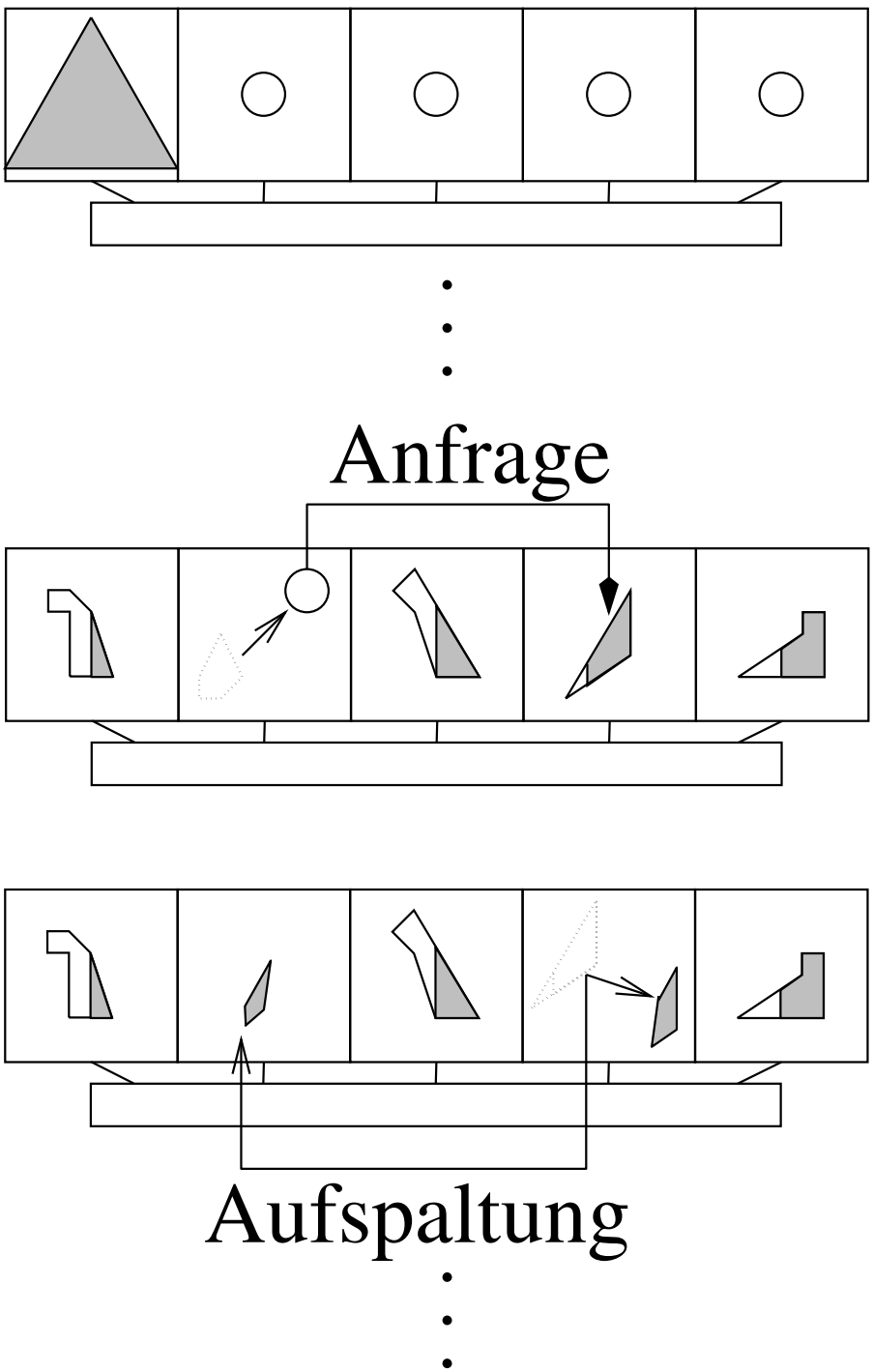
INFORMATIK

Receiver Initiated Load Balancing



INFORMATIK

Random Polling



$\tilde{\mathrm{O}}(\cdot)$ Calculus

$X \in \tilde{\mathrm{O}}(f(n))$ – iff $\forall \beta > 0$:

$$\exists c > 0, n_0 > 0 : \forall n \geq n_0 : \mathbf{Pr}[X > cf(n)] \leq n^{-\beta}$$

Advantage: simple rules for sum and maximum.

Termination Detection

Mattern's double counting method

Asynchronously compute global sum over

- **sent** and
- **received** messages.

But do not count requests and rejections. PEs send their values when

- They fall idle and
- have not contributed to the current cycle yet.

When **both** values are the same **twice** in a row global termination is announced. (Otherwise start new cycle).

Incurs delay in $\mathcal{O}(T_{\text{coll}})$.

Synchronous Random Polling

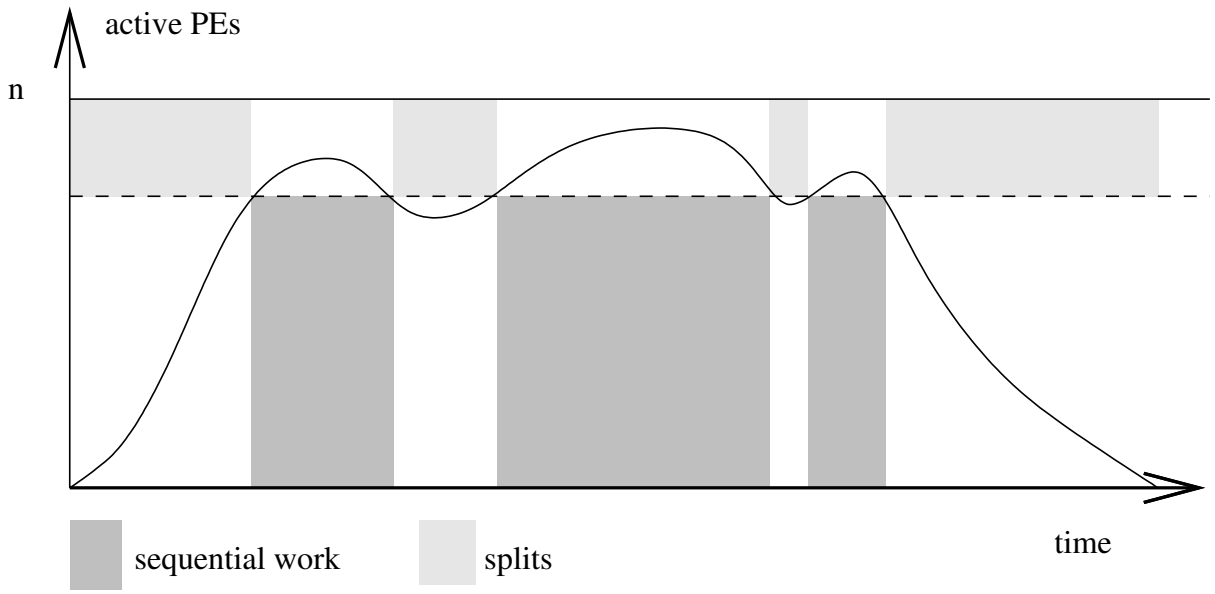
var P, P' : Subproblem
$P :=$ **if** $i_{\mathrm{PE}} = 0$ **then** P_{root} **else** $P_\emptyset$
loop
    $P := \mathrm{work}(P, \Delta t)$
    $m' := |\{i : T(P\langle i\rangle) = 0\}|$
    if $m' = n$ **then** exit loop
    elsif $m' \geq m$ **then**
        if $T(P) = 0$ **then**
            send a request to a random PE
        if there is an incoming request **then**
            $(P, P') := \mathrm{split}(P)$
            send P' to one of the requestors
            send empty subproblems the rest
        if $T(P) = 0$ **then** receive P

INFORMATIK

Analysis

Satz 3. *For all $\epsilon > 0$ there is a choice of Δt and m such that*

$$T_{\text{par}} \preceq (1+\epsilon)\frac{T_{\text{seq}}}{n} + \tilde{\text{O}}\left(T_{\text{atomic}} + h(T_{\text{rout}}(l) + T_{\text{coll}} + T_{\text{split}})\right) \quad .$$

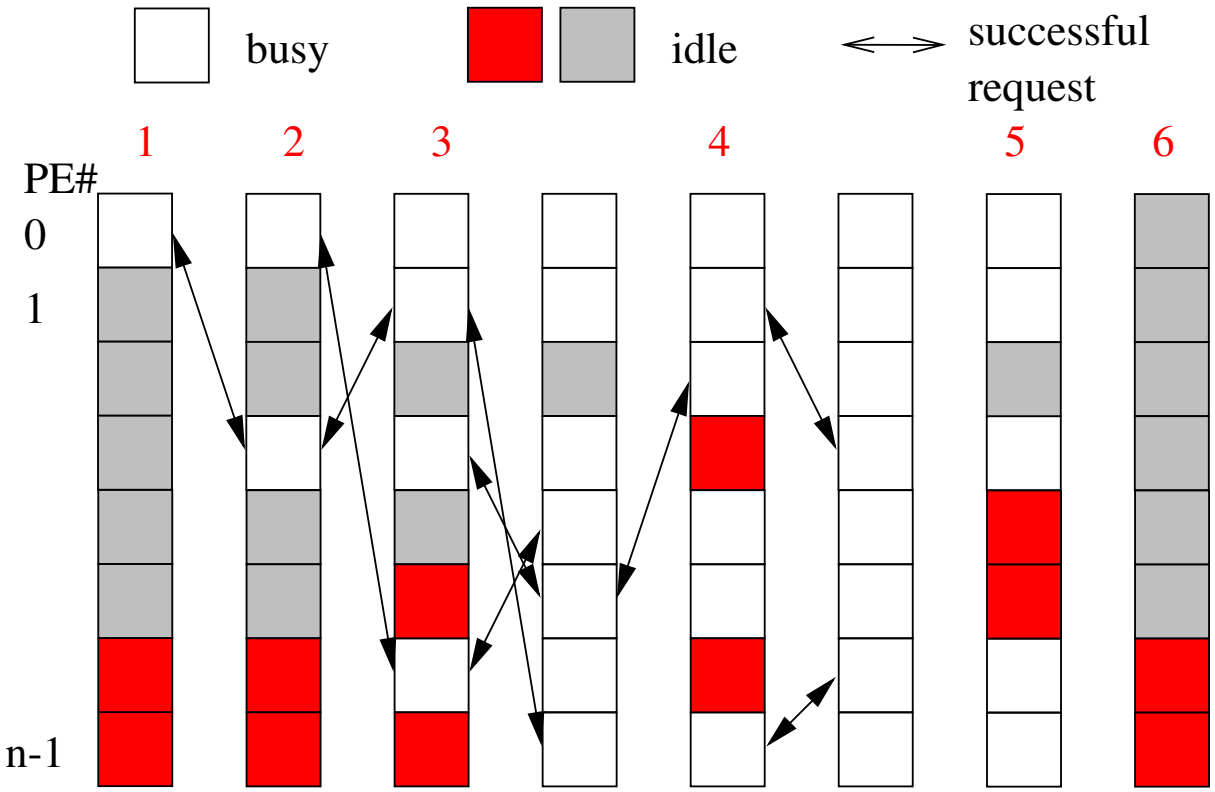


INFORMATIK

Bounding Idleness

Lemma 4. *Let $m < n$ with $m \in \Omega(n)$. Then $\tilde{\mathrm{O}}(h)$ iterations with at least m empty subproblems suffice to ensure*

$$\forall P : \mathrm{gen}(P) \geq h \ .$$



Busy phases

Lemma 5. *There are at most* $\frac{T_{\text{seq}}}{(n-m)\Delta t}$ *iterations with* $\leq m$ *idle PEs at their end.*

INFORMATIK

A Simplified Algorithm

var P, P' : Subproblem
$P :=$ **if** $i_{\mathrm{PE}} = 0$ **then** P_{root} **else** $P_\emptyset$
while not finished
    $P := \mathrm{work}(P, \Delta t)$
    select a global value $0 \leq s < n$ uniformly at random
    if $T(P\langle i_{\mathrm{PE}} - s \bmod n\rangle) = 0$ **then**
        $(P, P\langle i_{\mathrm{PE}} - s \bmod n\rangle) := \mathrm{split}(P)$

Satz 6. *For all $\epsilon > 0$ there is a choice of Δt and m such that*

$$T_{\mathrm{par}} \preceq (1+\epsilon)\frac{T_{\mathrm{seq}}}{n} + \tilde{\mathrm{O}}\left(T_{\mathrm{atomic}} + h(T_{\mathrm{rout}}(l) + T_{\mathrm{split}})\right) \quad .$$

INFORMATIK

Asynchronous Random Polling

var P, P' : Subproblem
$P :=$ **if** $i_{\mathrm{PE}} = 0$ **then** P_{root} **else** $P_\emptyset$
while no global termination yet **do**
    if $T(P) = 0$ **then**
        send a request to a random PE
    else
        $P := \mathrm{work}(P, \Delta t)$
    if there is an incoming message M **then**
        if M is a request from PE j **then**
            $(P, P') := \mathrm{split}(P)$
            send P' to PE j
        else
            $P := M$

Analysis

Satz 7.

$$\mathbf{E}T_{\text{par}} \leq (1+\epsilon)\frac{T_{\text{seq}}}{n} + \mathcal{O}(T_{\text{atomic}} + h\left(\frac{1}{\epsilon} + T_{\text{rout}} + T_{\text{split}}\right))$$

for an appropriate choice of Δt*.*

- $\tilde{\text{O}}$ postponed for now
- Exploit that LogP model is asynchronous
- Iteration counting $\rightsquigarrow$ Operation counting

Simple Observations

Lemma 8.

1. *The total amount of active CPU work expended for processing a request is bounded by* $T_{\text{split}} + \mathcal{O}(T_{\text{rout}})$.
2. *If any requests have arrived at a PE, at least one of the requests is answered every* $\Delta t + T_{\text{split}} + \mathcal{O}(T_{\text{rout}})$ *time units.*
3. *The expected elapsed time between the arrival of a message and sending the corresponding reply is in* $\mathcal{O}(\Delta t + T_{\text{split}} + T_{\text{rout}})$.

INFORMATIK

Accounting for Dead Times

Lemma 9. *All dead times can be covered by associating a dead time* $T_{\text{tot}} = \Delta t + T_{\text{split}} + \mathcal{O}(T_{\text{rout}})$ *with each request reaching a subproblem.*

Seeing Red

Definition 10. *Red* request $\Longrightarrow \leq n$ other red requests T_{tot} after its arrival.

Lemma 11. $\forall \beta > 0 : \exists$ *constant* $c > 0 :$ *after processing cnh red requests*
$\mathbf{Pr}[\exists i : \text{gen}(I\langle i \rangle) < h] \leq n^{-\beta}$

Lemma 12. *Requests can be colored in such a way that an expected work in* $\mathcal{O}(hn(\Delta t + T_{\text{split}} + T_{\text{rout}})$ *for all request processing suffices to process chn red requests.*

Lemma 13. *The expected overall expense for communicating, splitting and waiting until there are no more subproblems with* $\text{gen}(P) < h$ *is in* $\mathcal{O}(hn(\Delta t + T_{\text{split}} + T_{\text{rout}}))$.

Choosing Δt

- $\Delta t \in \mathcal{O}(T_{\text{rout}} + T_{\text{split}})$
- $\Delta t \in \Omega\big(\min\left(\frac{T_{\text{atomic}}}{h}, T_{\text{rout}} + T_{\text{split}}\right)\big)$
- $\Delta t > T_{\text{poll}}/\epsilon$

- Vain polling must not eat up too much time.
- Finishing up atomic pieces must make progress in the presence of many requests.
- Termination detection messages must get through.

INFORMATIK

Interpretation

- $T_{\text{atomic}}/h \leq h(T_{\text{rout}} + T_{\text{split}})$.
- Arbitrarily fine granularity OK.
- $T_{\text{seq}}/n \gg h(T_{\text{rout}} + T_{\text{split}})$.

INFORMATIK

A Trivial Lower Bound

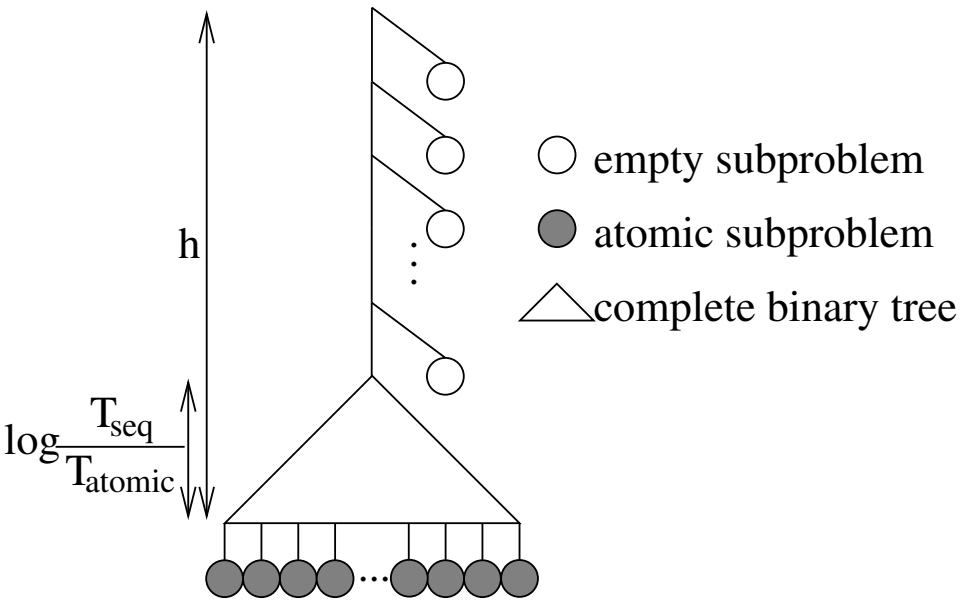
Satz 14. *For* **all** *tree shaped computations*

$$T_{\text{par}} \in \Omega\left(\frac{T_{\text{seq}}}{n} + T_{\text{atomic}} + T_{\text{coll}} + T_{\text{split}} \log n\right) \quad .$$

if efficiency in $\Omega(1)$ *shall be achieved.*

INFORMATIK

Many Consecutive Splits



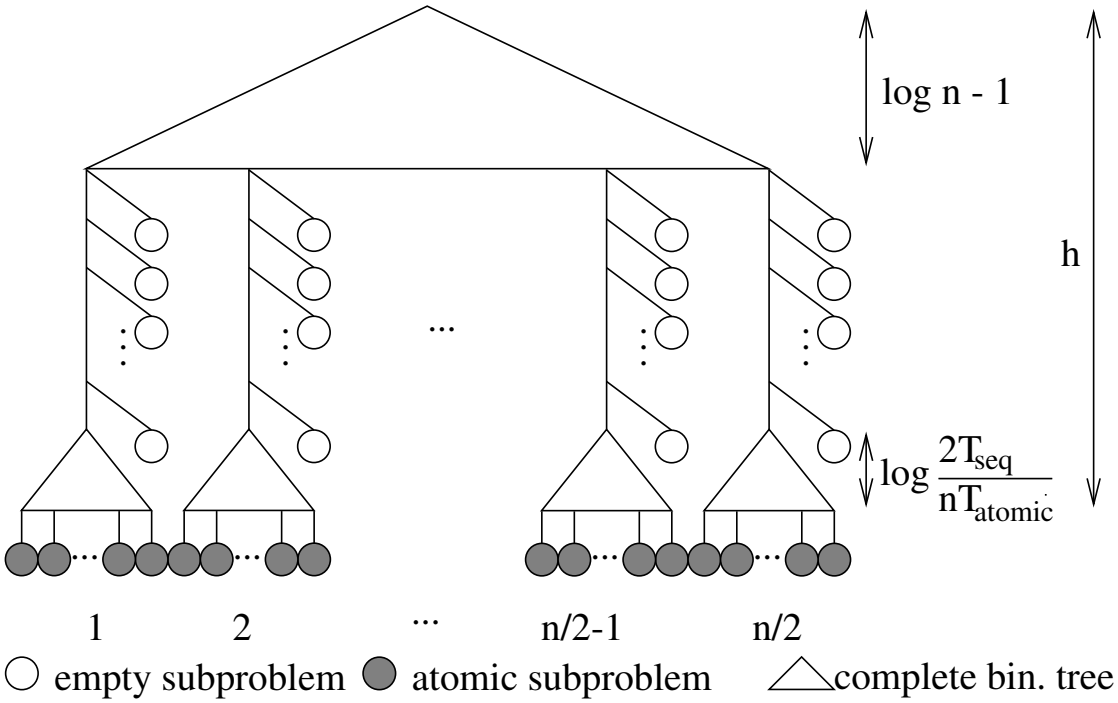
Additional

$$h - \log \frac{T_{\text{seq}}}{T_{\text{atomic}}}$$

term.

INFORMATIK

Many Splits Overall



Satz 15. *Some problems need at least*

$$\frac{n}{2}\left(h - \log\frac{T_{\text{seq}}}{T_{\text{atomic}}}\right)$$

splits for efficiency $\geq \frac{1}{2}$.

Korollar 16. *Receiver initiated algorithms need a corresponding number of communications.*

Satz 17 (Wu and Kung 1991). *A similar bound holds for all deterministic load balancers.*

The Initial Phase of Random Polling

Consider the simple synchronous random shift algorithm:

- Synchronous random shift algorithm
- $\Delta t = 0$
- root problem at random PE
- $X_{ik} :=$ [PE i busy before phase k]

"Heuristic" Analysis

$$\mathbf{Pr}[X_{i,k+1}=0] \stackrel{?}{=} \mathbf{Pr}[X_{ik}=0]\sum_{j\neq i}\frac{1}{n-1}\mathbf{Pr}[X_{jk}=0]$$
$$= \mathbf{Pr}[X_{ik}=0]^2 \ .$$

This would imply

$$\mathbf{Pr}[X_{ik}=0] = (1-1/n)^{2^k} \leq e^{-2^k/n}$$

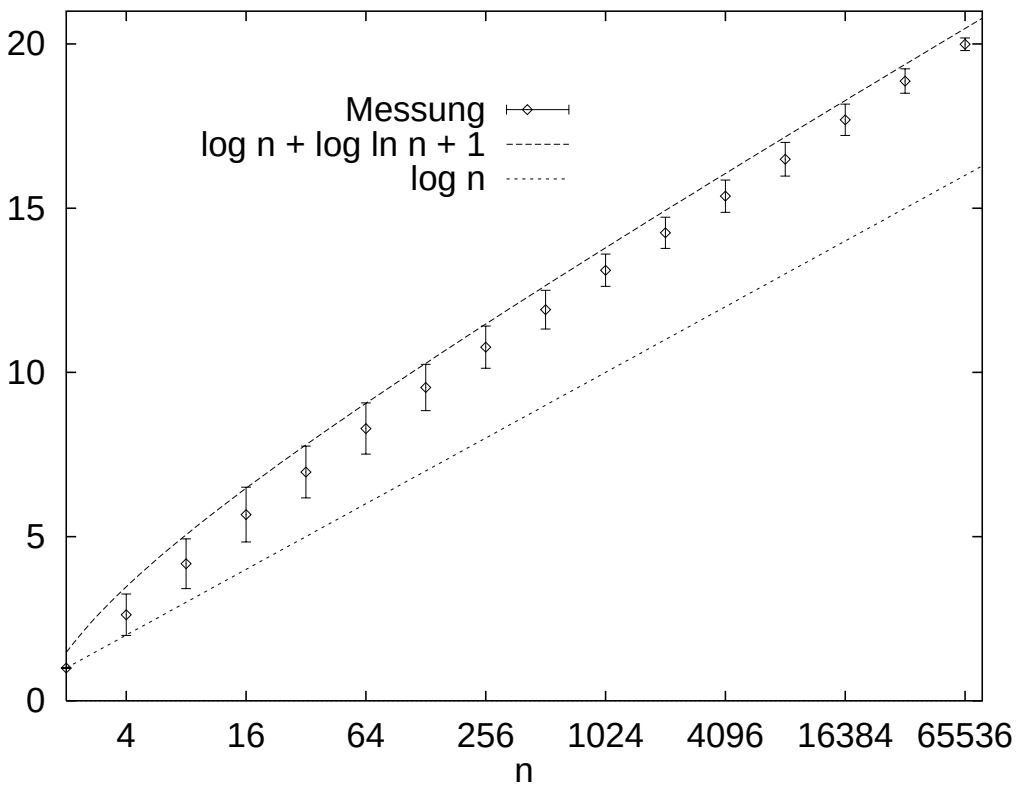
and at $k = \log n + \log\ln n$

$$\mathbf{E}\#\text{busy PEs} = n-1$$

Vermutung 18. $\log n + \log\ln n + 1$ *phase suffice to get all PEs busy.*

INFORMATIK

Experimental Validation

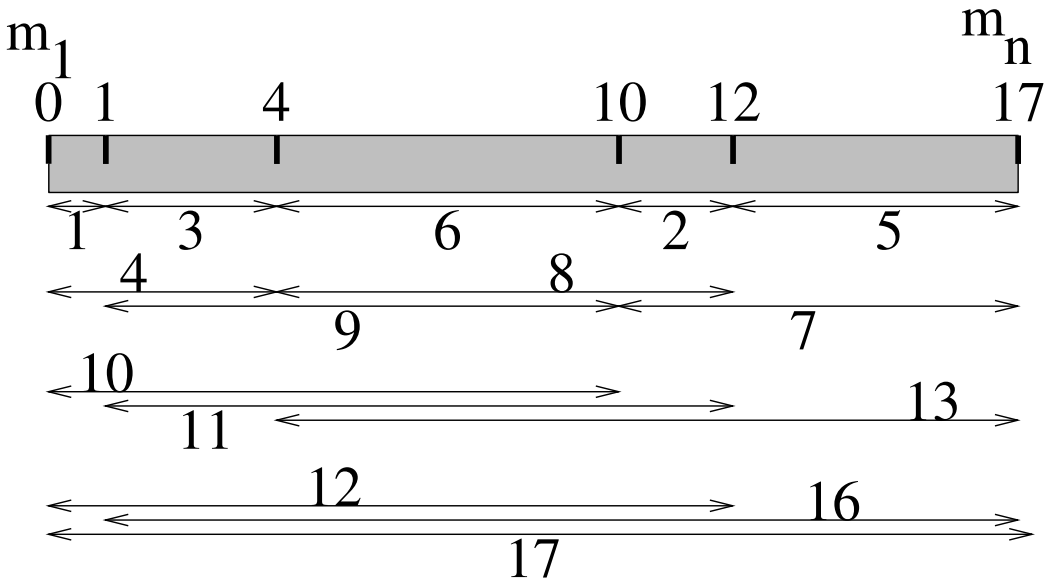


- 1000 repetitions
- Standard deviation small
- Student t-test yields numeric underflow

Golomb-Rulers

- Total length m
- find n marks $\{m_1, \ldots m_n\} \subseteq \mathbb{N}_0$
- $m_1 = 0$, $m_n = m$
- $|\{m_j - m_i : 1 \leq i < j \leq n\}| = n(n-1)/2$

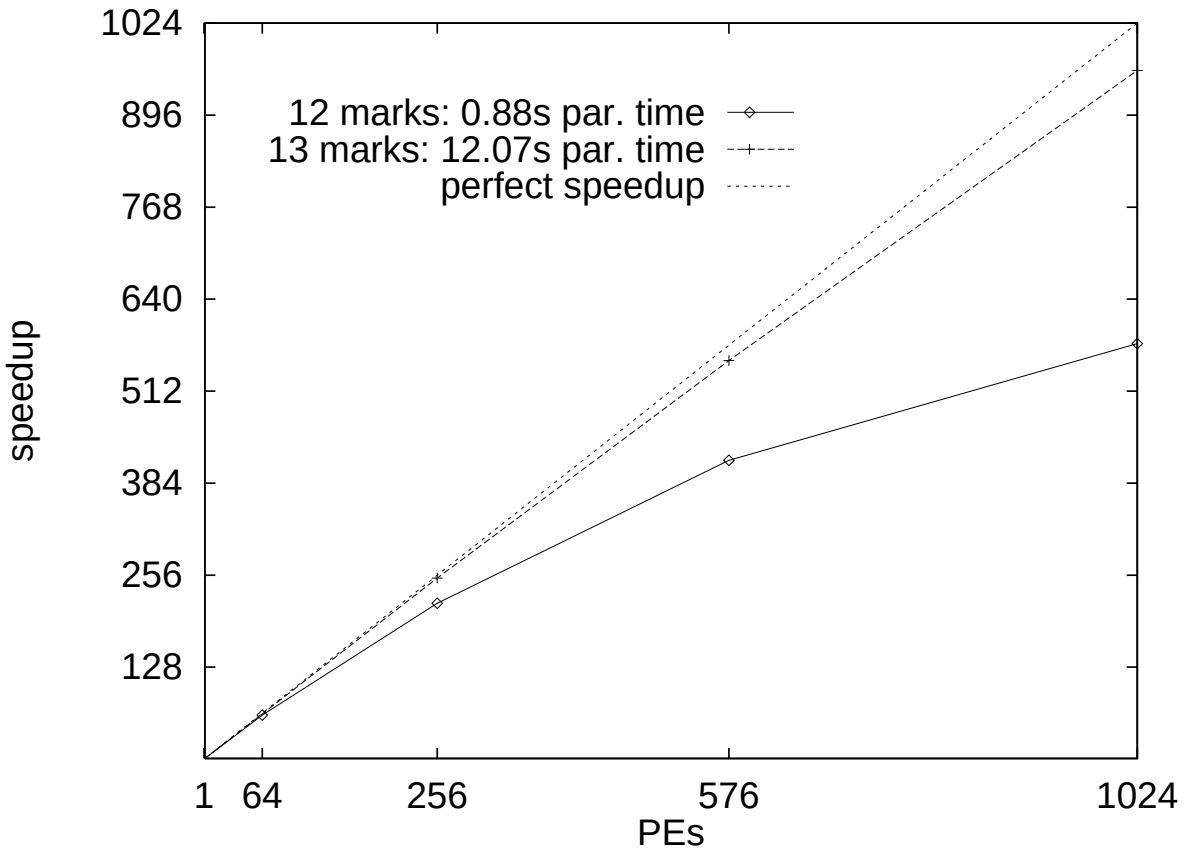
Applications: Radar astronomy, codes, ...



INFORMATIK

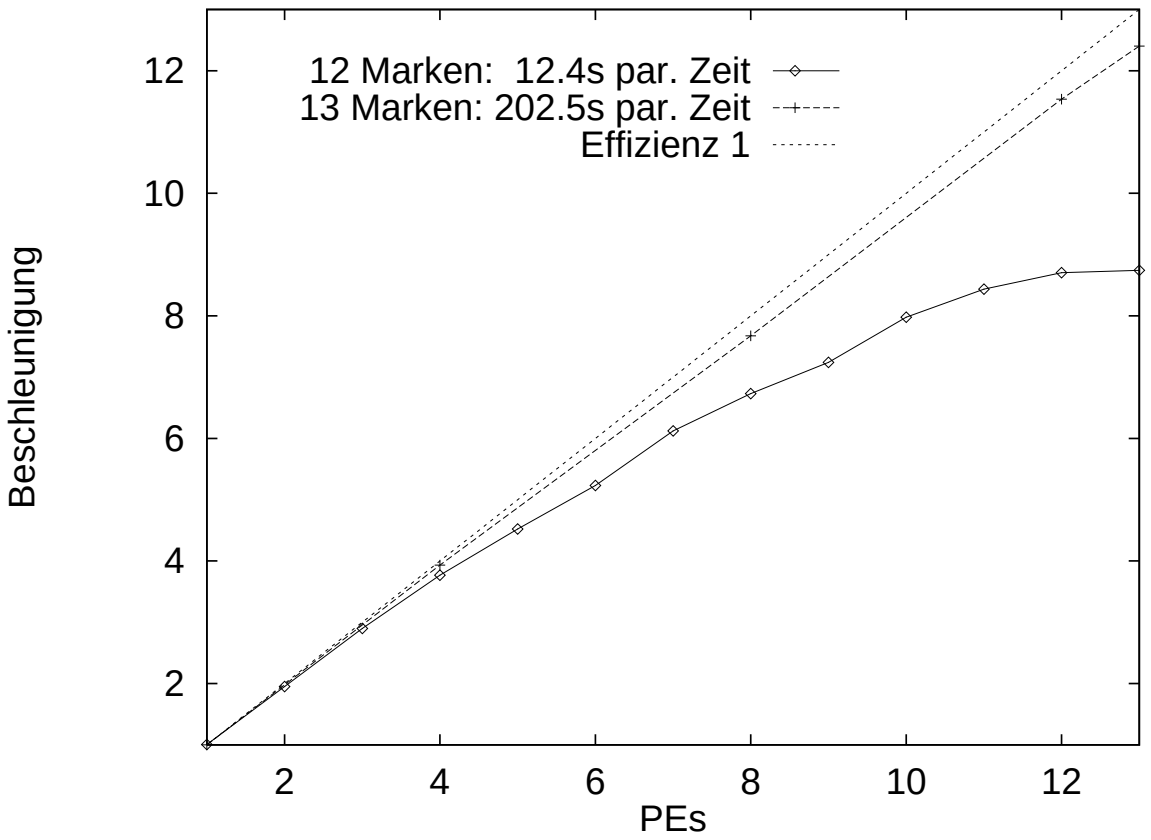
Many Processors

- Parsytec GCel-3/1024 with Cosy (PB)
- Verfication search



INFORMATIK

LAN



- Differing PE-Speeds (even dynamically) are unproblematic.
- Even complete suspension is OK as long as requests are answered.

INFORMATIK

The 0/1-Knapsack Problem

- m items
- maximum knapsack weight M
- item weights w_i
- item profits p_i
- Find $x_i \in \{0, 1\}$ such that
  - $\sum w_i x_i \leq M$
  - $\sum p_i x_i$ is maximized

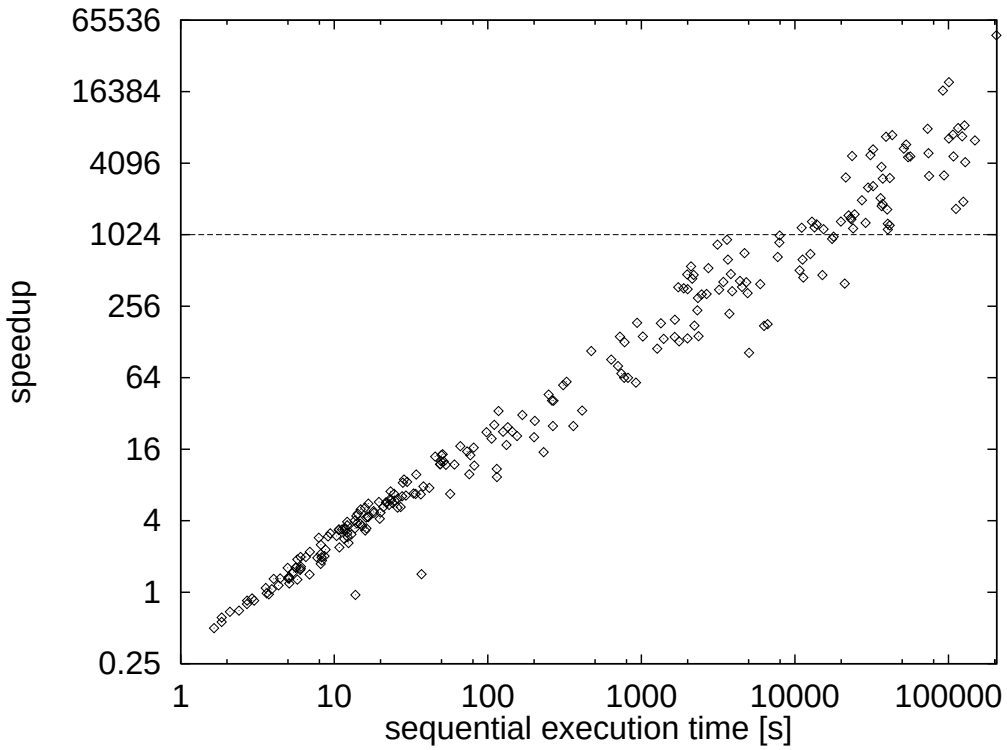
Best known approach for large m:

- Depth-first branch-and-bound
- Bounding function based on a the relaxation $x_i \in [0, 1]$. (Can be computed in $\mathcal{O}(\log m)$ steps.)

INFORMATIK

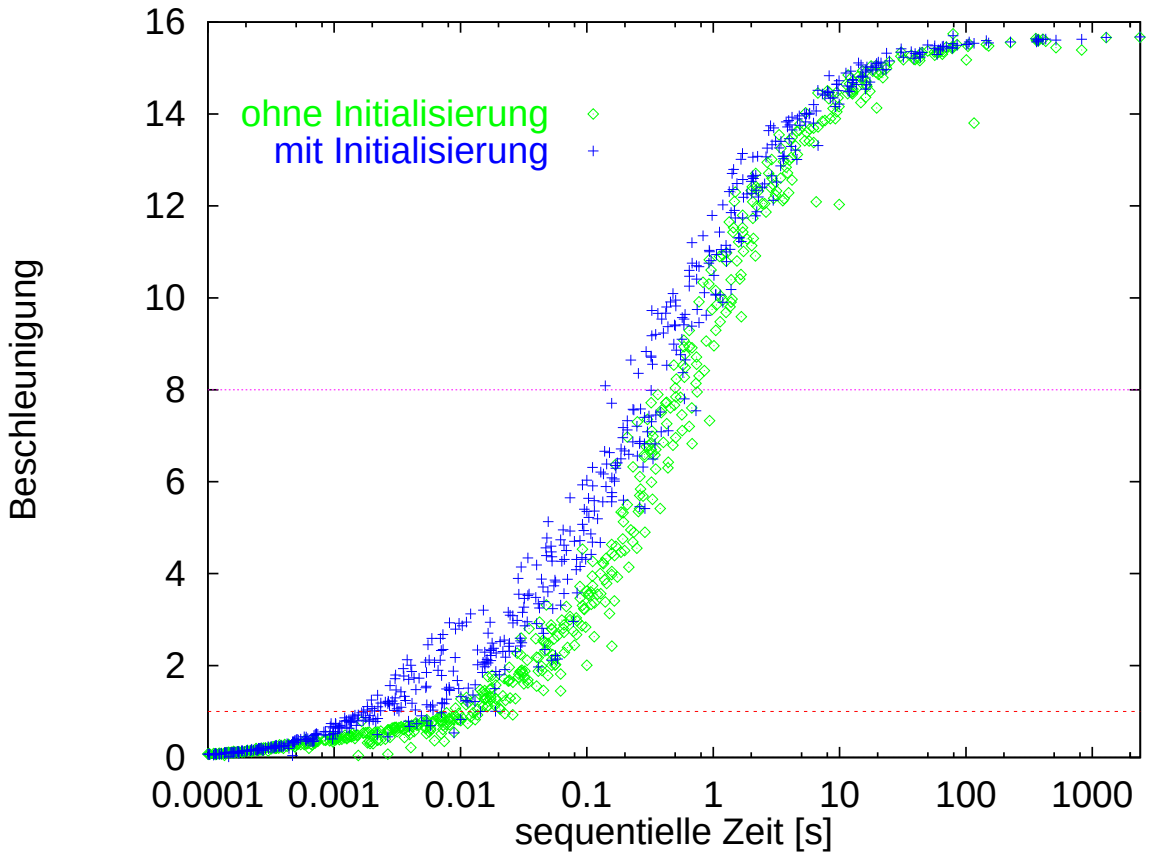
Superlinear Speedup

- Parsytec GCel-3/1024 under Cosy (PB)
- 1024 processors
- 2000 items
- Splitting on all levels
- 256 random instances at the border between simple and difficult
- overall 1410× faster than seq. computation!



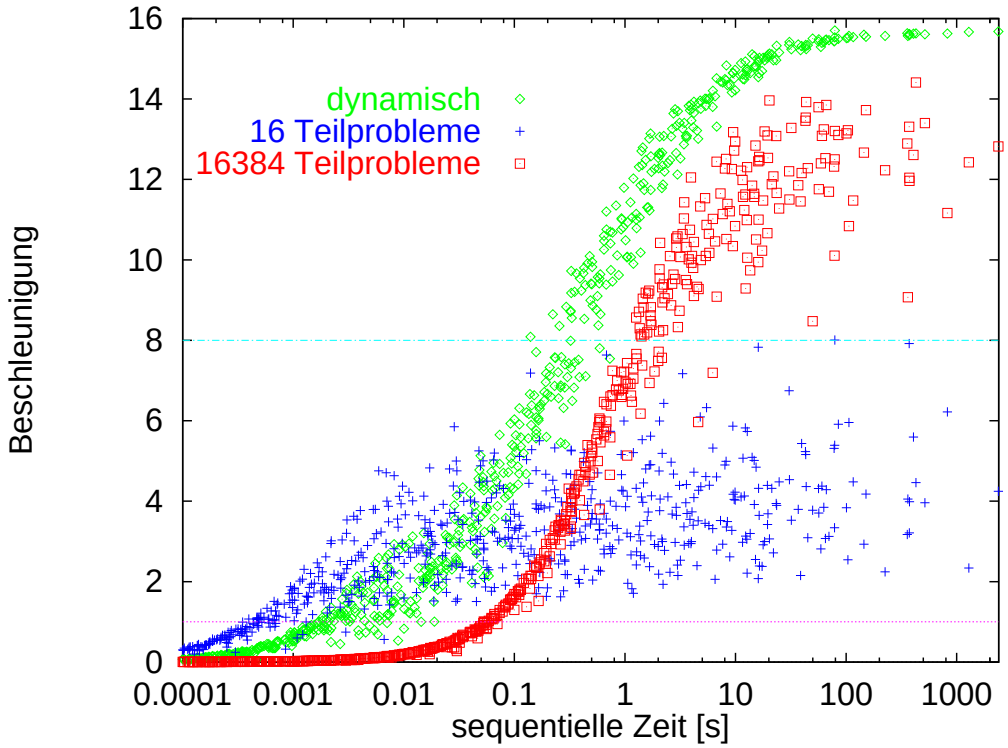
INFORMATIK

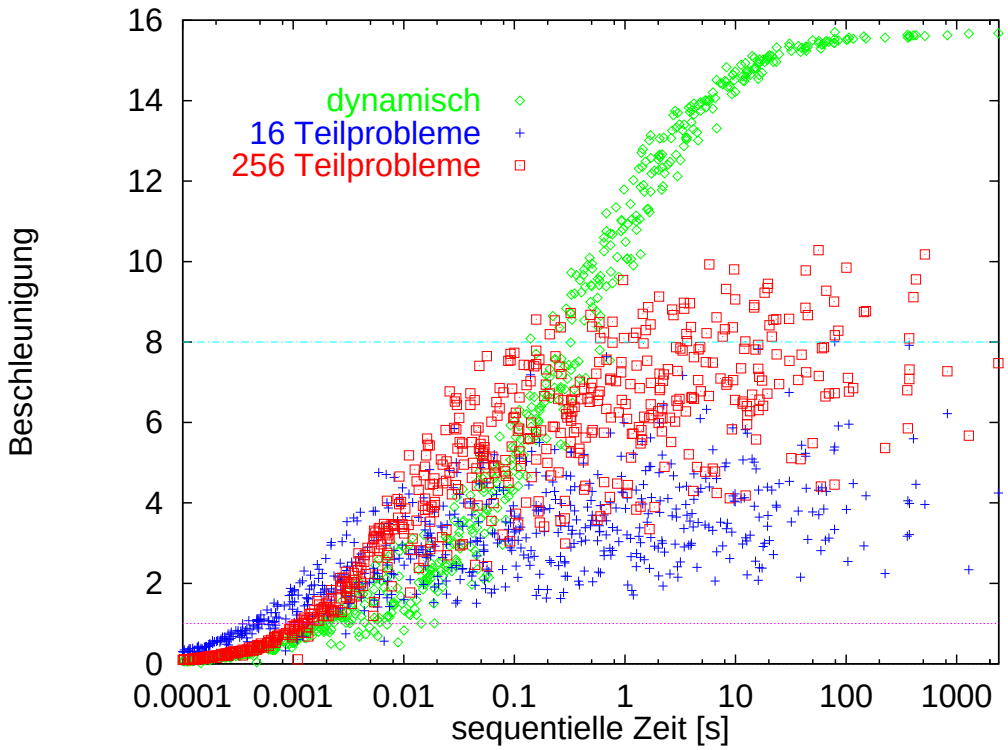
Fast Initialization



INFORMATIK

Static vs Dynamic LB

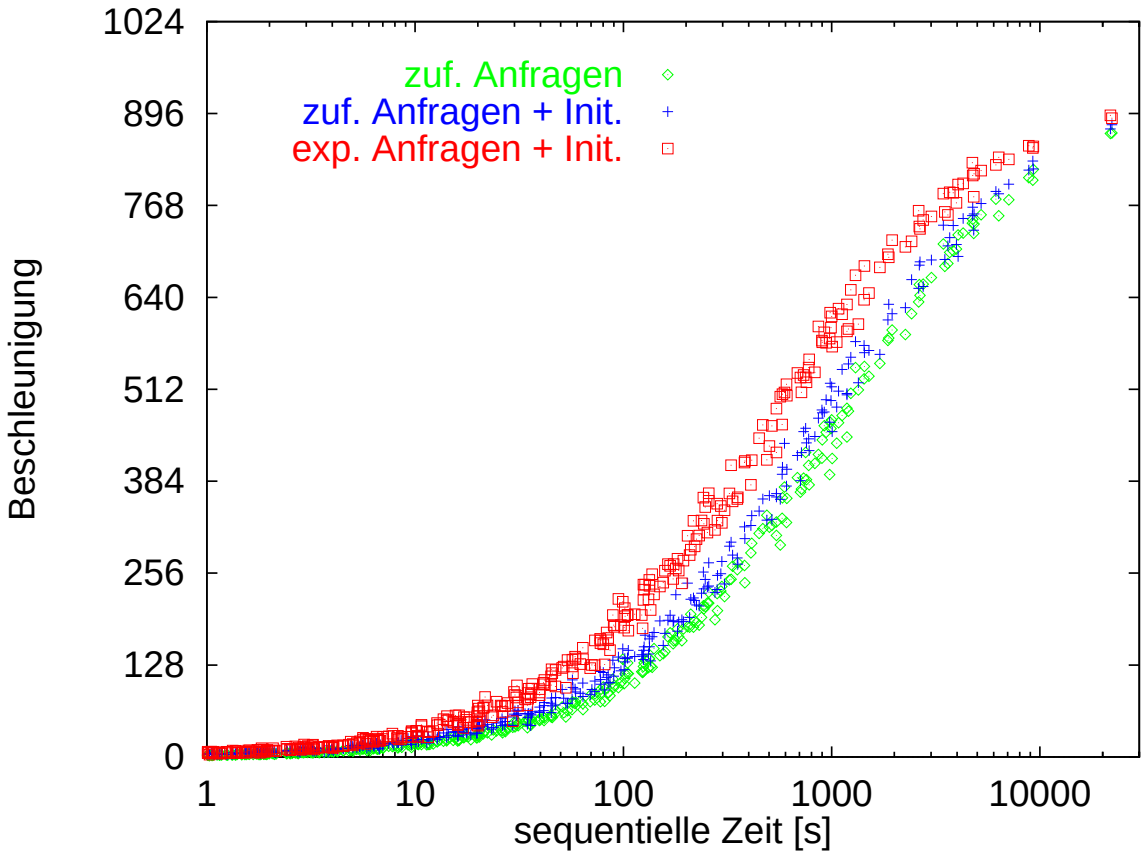




INFORMATIK

Beyond Global Polling

- Randomized Initialization
- Asynchronously increase polling range (exponentially)



Game Tree Search

- Naive Parallelization yields only Speedup $\mathcal{O}(\sqrt{n})$.
- Young Brother Wait Concept (Feldmann et al.)
- Tradeoff between Speculativity and Sequentialization
- Propagate window updates
- Combine with global transposition table