Parallel scaling of Teter's Minimization for *Ab Initio* Calculations

Torsten Hoefler

Department of Computer Science
Technical University of Chemnitz

HPCNano Workshop 2006
Supercomputing'06

Tampa, FL, USA

November 13th 2006

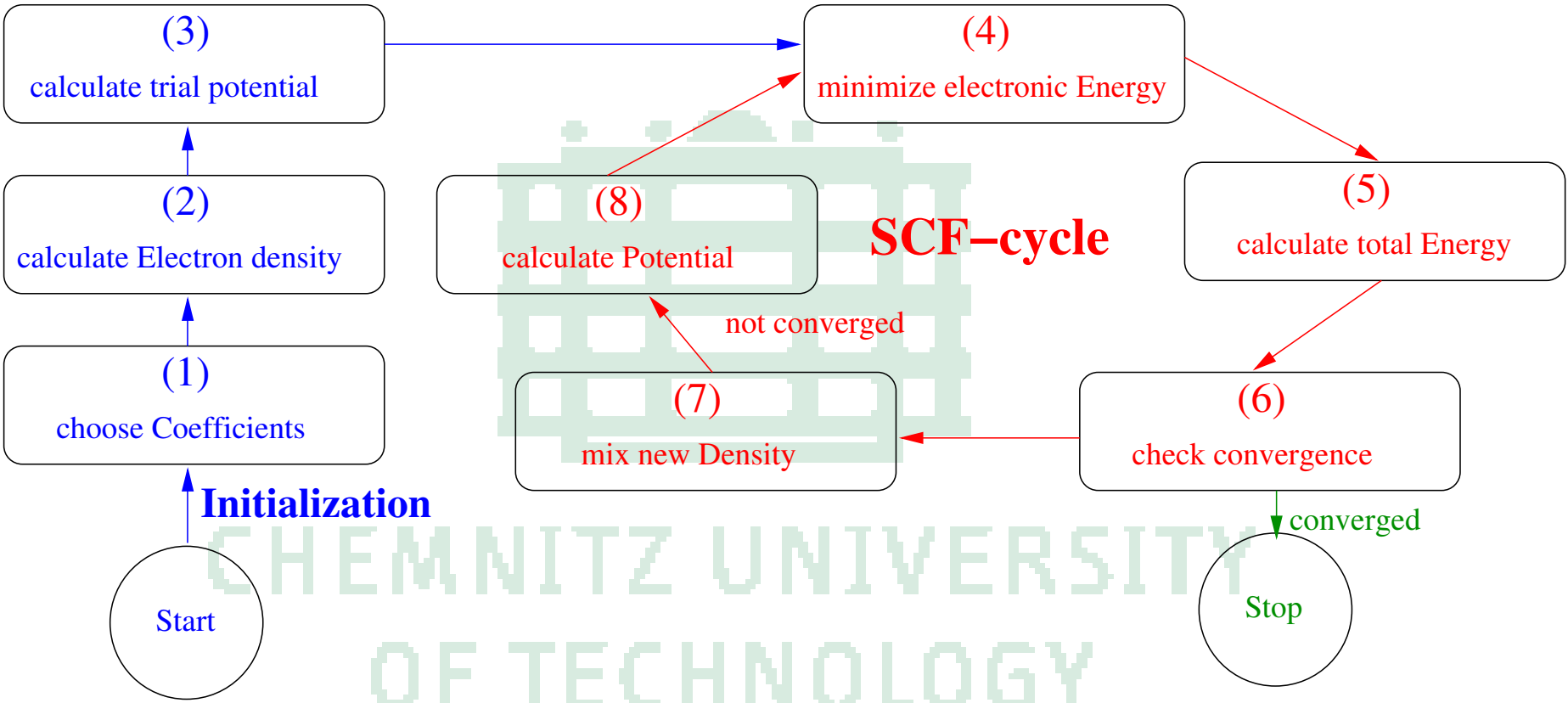
Outline

Outline

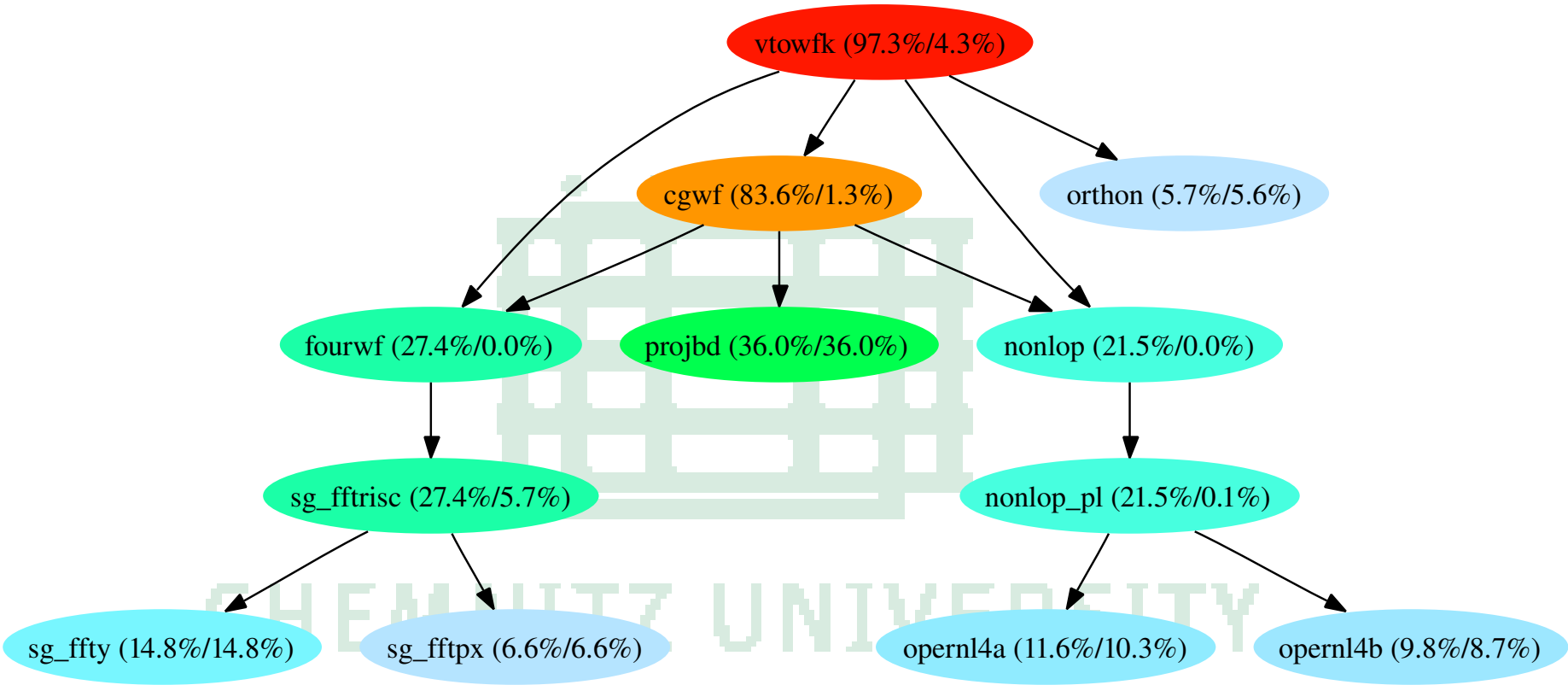
ABINIT Introduction

- ABINIT solves time-independent Schrödinger equation
- effective one-particle case, uses DFT
- $\widehat{H}_{\text{tot}}\Phi = E_{\text{tot}}\Phi$
- $\Rightarrow$ Eigenvalue problem
- Eigenvalues and -vectors determined with CG minimization (Teter et al.)
- wavefunction Φ written in plain-wave basis set

ABINIT Program Flow



ABINIT Tracing



$\Rightarrow$ 83% for Teter minimization

Outline

Conjugate Gradient Operations

- dot- and matrix-vector product
- dot-product: $\langle \Phi_i | \Phi_j \rangle$
- matrix-vector product: $\widehat{H}\Phi$
- $\widehat{H} = E^e_{kin} + V^e_{loc} + V^e_{nl}$
- E^e_{kin} and V^e_{loc} in reciprocal (k-) space
- V^e_{nl} in real space
- $\Rightarrow$ 3D-FFT to transform between real and reciprocal space

Outline

K-Point Parallelization

- Bands have to be minimized for each k-point
- Minimization for each k-point is independent
- All k-point data is only needed for the calculation of ETOT
- $\Rightarrow$ straightforward parallelization
- ABINIT implementation:
  - Good speedup :-)
  - Uses only collective communication :-)
  - Limited to *nkpt* :-(
  - Uses MPI_COMM_WORLD :-(
  - Uses MPI_BARRIER :-(

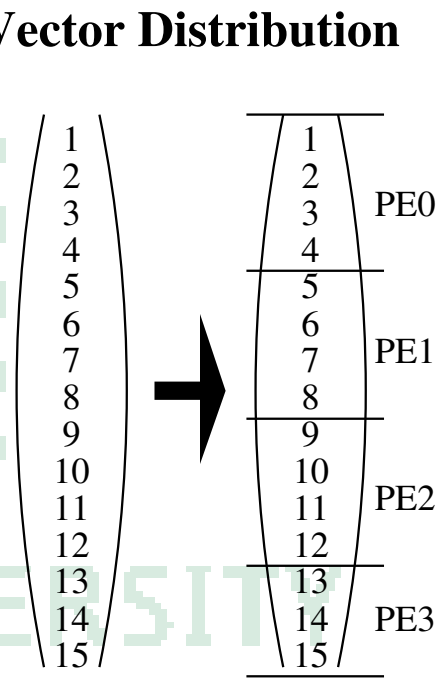
Band Parallelization

- The Teter Method allows parallel CG
- Orthogonalization constraint forces non-ideal solution
- $\Rightarrow$ tricky parallelization
- ABINIT implementation:
  - Speedup depends on interconnect :-/
  - Uses Send/Recv :-(
  - Limited by $nband/c$ (c not easily predictable)

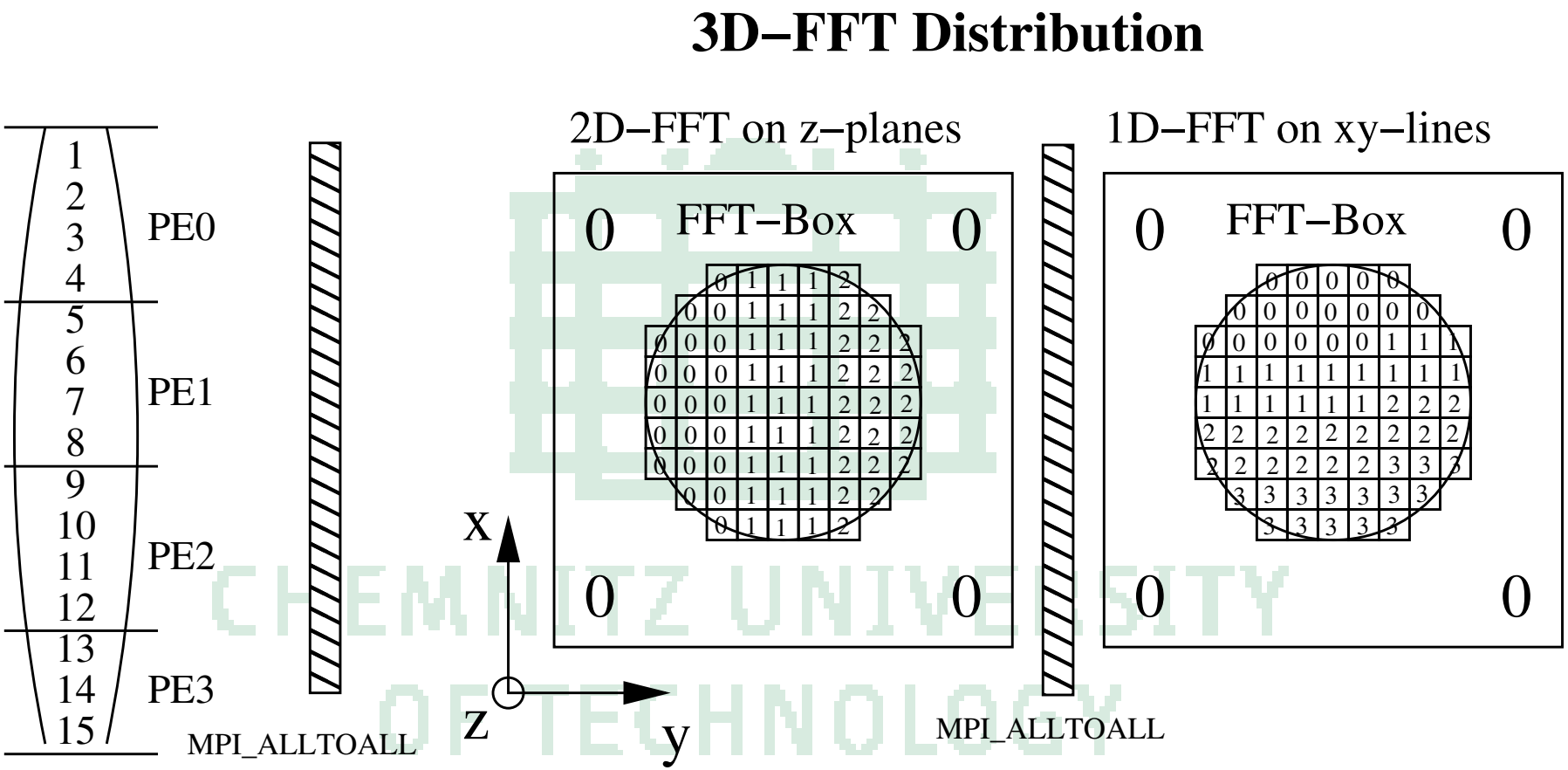
Outline

G Parallelization

- FFT $\Rightarrow$ Two parallelization schemes:
  - Distribute plane wave coefficients
  - Distribute real space FFT Grid
- Strict load balancing
- Minimize communication
- Possible to combine with Band and k-Point parallelization



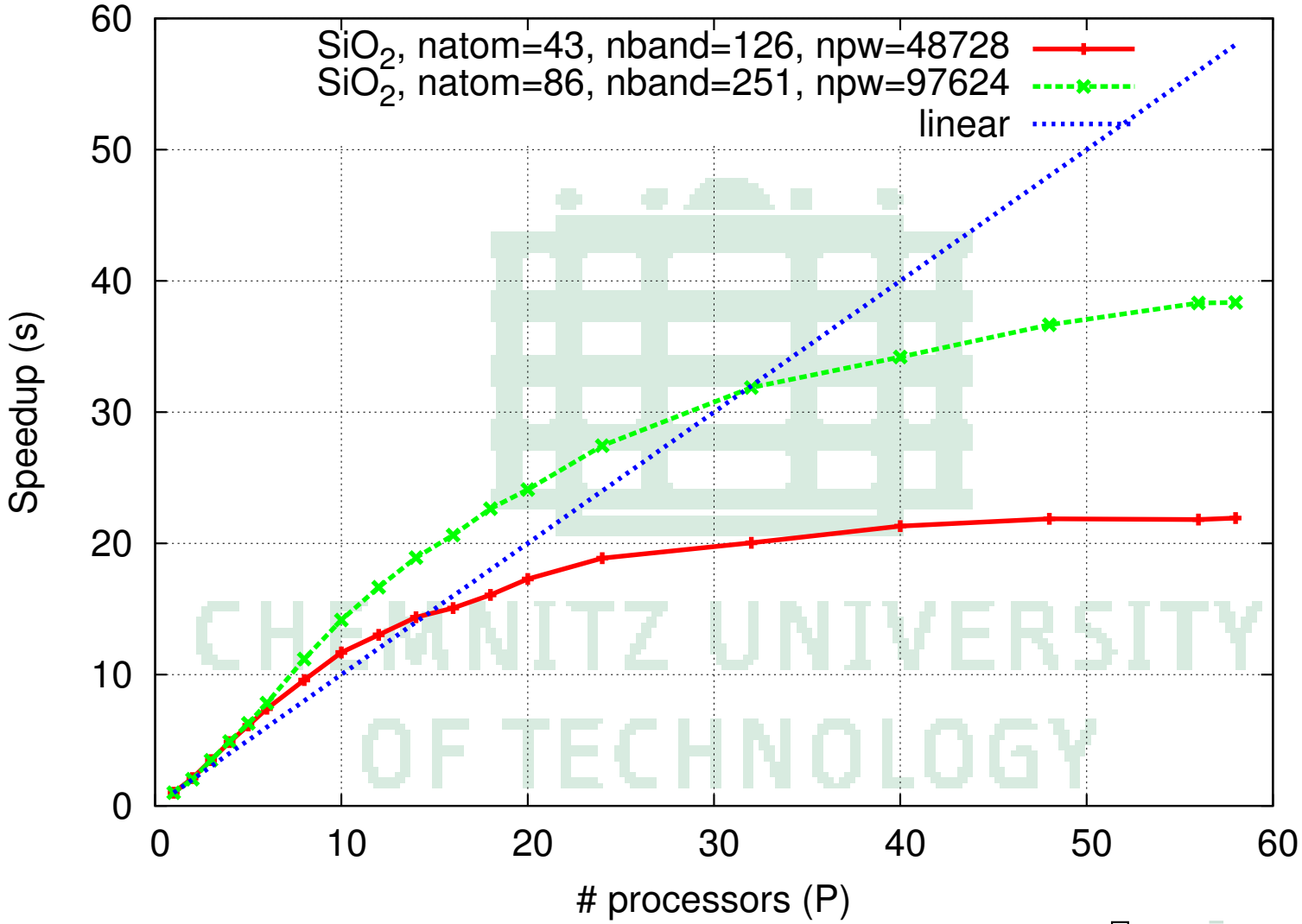
Real Space Distribution



Implementation Issues

- Necessary communication (complexity):
  - Dot-products ($O(1)$)
  - Computation of kinetic energy ($O(1)$)
  - FFT transpose ($O(natom)$)
- Only collective communication:
  - MPI_ALLREDUCE for reductions
  - MPI_ALLTOALL for FFT transpose
- Principles:
  - only coll. communication
  - separate communicator
  - simplification of the main code
  - heavy usage of math librarys

Benchmarking the Implementation of `cgwf`

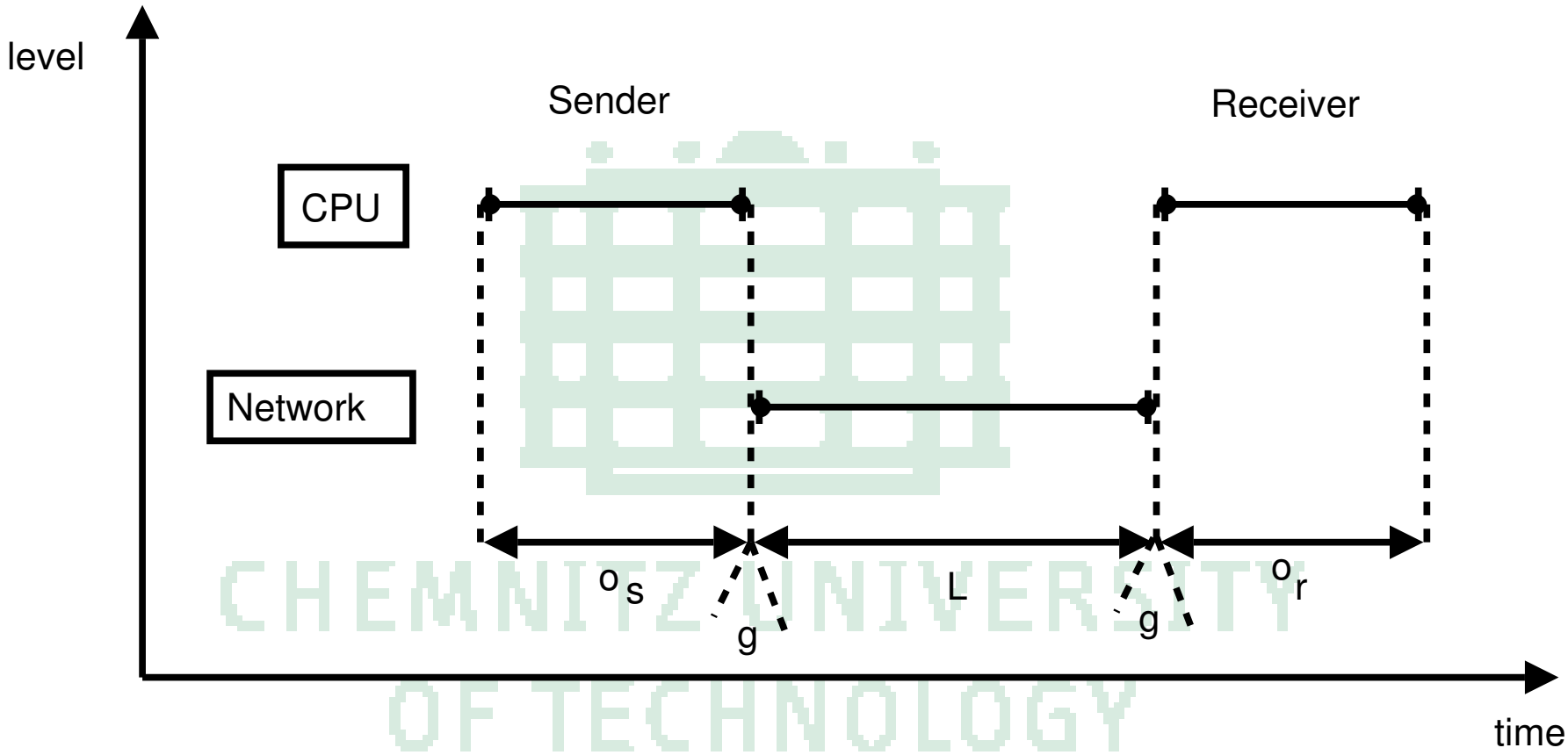


Possible Reasons for limited Scalability

- serial parts (Amdahl's law)
  - allocations
  - scalar calculation
  - index reordering (packin,packout - FFT)
- communication overhead
  - latency of blocking collective operations
  - limits scalability significantly
  - overhead will be modelled in the following

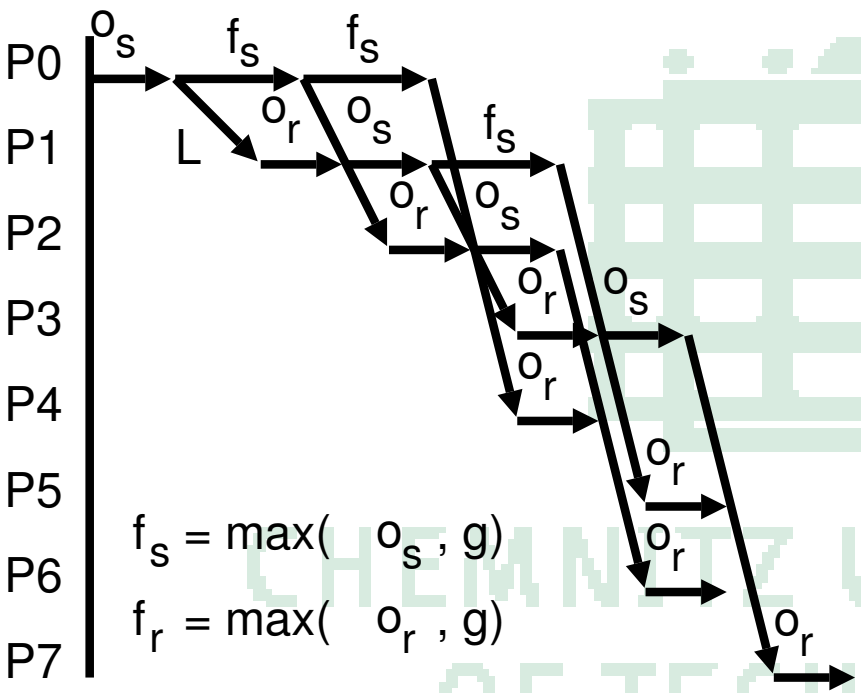
Outline

The LogP Model



Modelling the MPI_ALLREDUCE

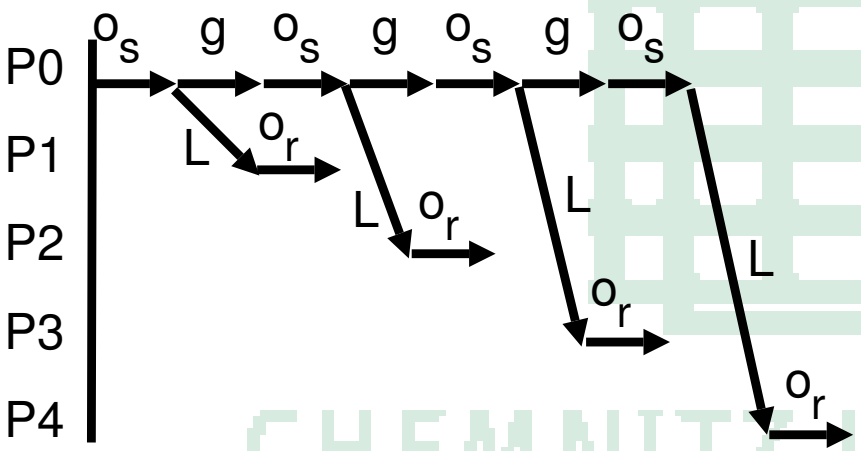
- $\rightarrow$ MPI_REDUCE to node 0 and MPI_BCAST



- $t_{red}(P, size) = 2 \cdot size \cdot (2o + L + (\lceil log_2 P \rceil - 1) \cdot max\{g, 2o + L\})$

Modelling the MPI_ALLTOALL

- $\longrightarrow$ each node hast to send to all others
- single host:



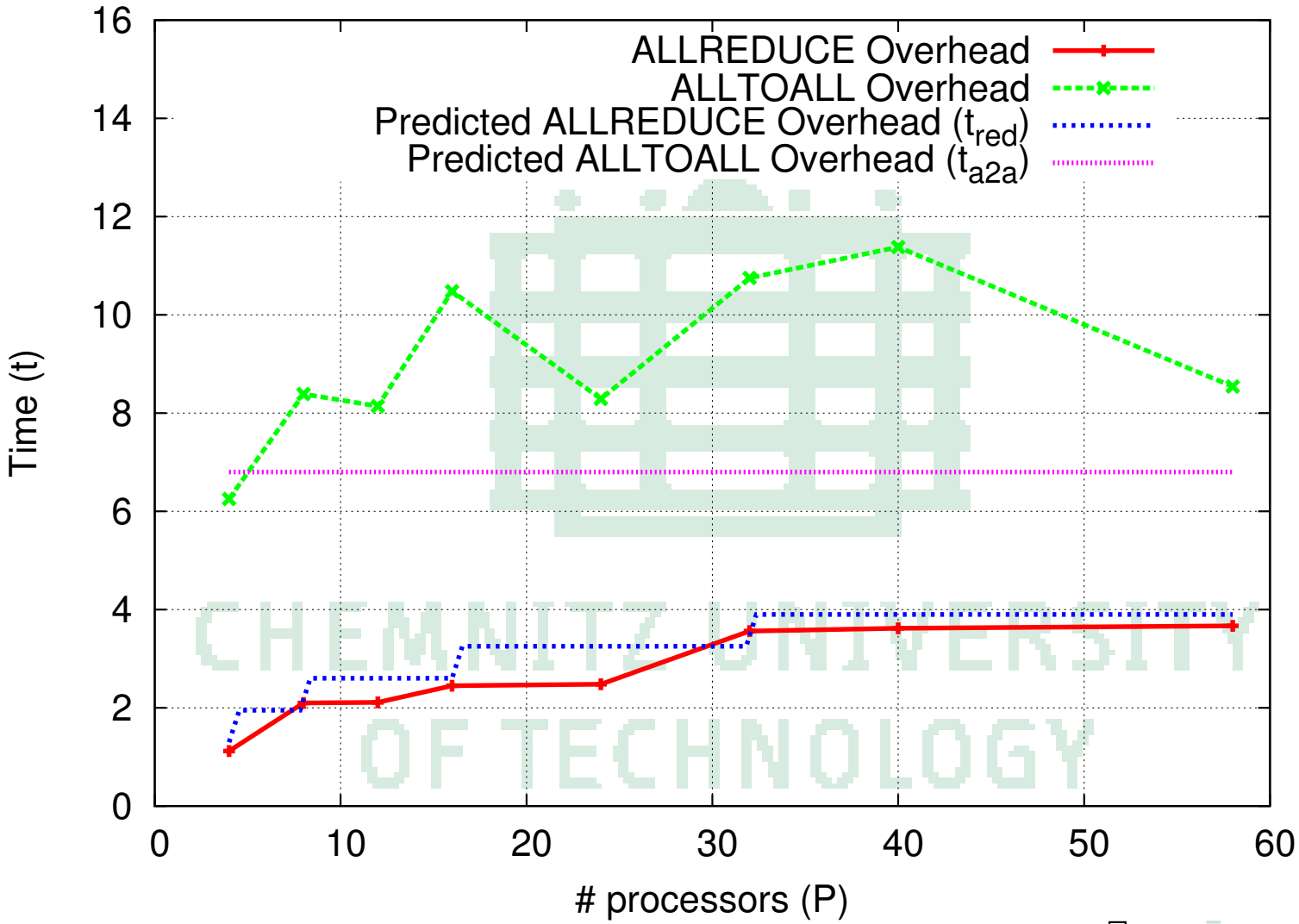
- all hosts send, assuming FBB
- $t_{a2a}(P, size) = size \cdot ((2o + L) + (P - 1) \cdot (g + o))$

Predicting the Overhead

- $o_{red}(P) = nband \cdot (9 + 2 \cdot nband) \cdot t_{red}(P, 1)$
- $o_{red}(P) = O(log_2 P)$
- $natom = 43$:
  - $o_{red}(P) = 126 \cdot (9 + 2 \cdot 126) \cdot 2 \cdot (\lceil log_2 P \rceil \cdot 9.88)$
  - $o_{red}(P) = 65772 \cdot (\lceil log_2 P \rceil \cdot 9.88)$

- $o_{a2a}(P) = 2 \cdot o_{a2a}(P, N_x \cdot N_y \cdot N_z / P)$
- $o_{a2a}(P) = O(1)$
- $natom = 43$:
  - $o_{a2a}(P) = \ldots$
  - $o_{a2a}(P) \approx 6.3s$

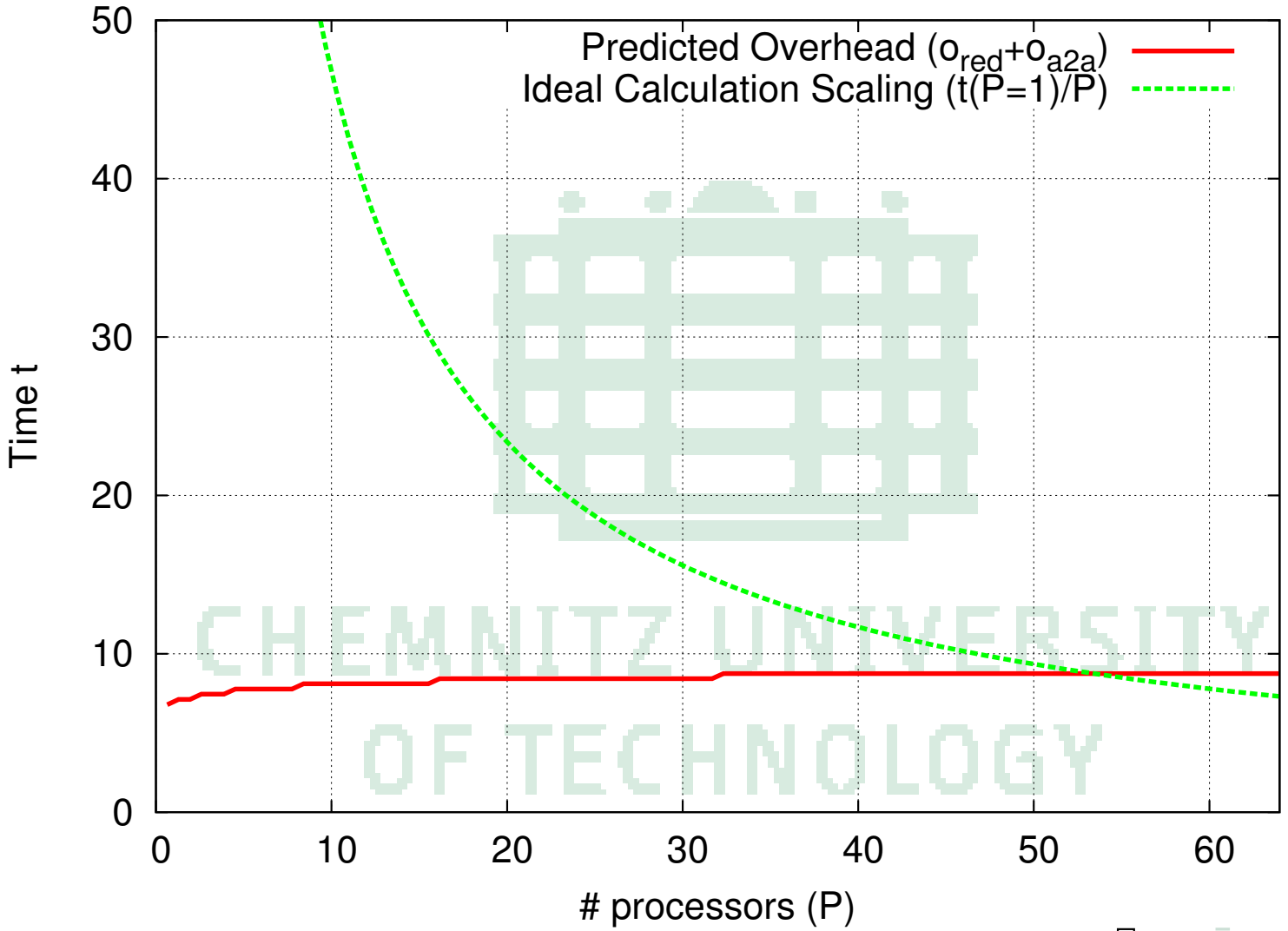
Verifying the Overhead Prediction



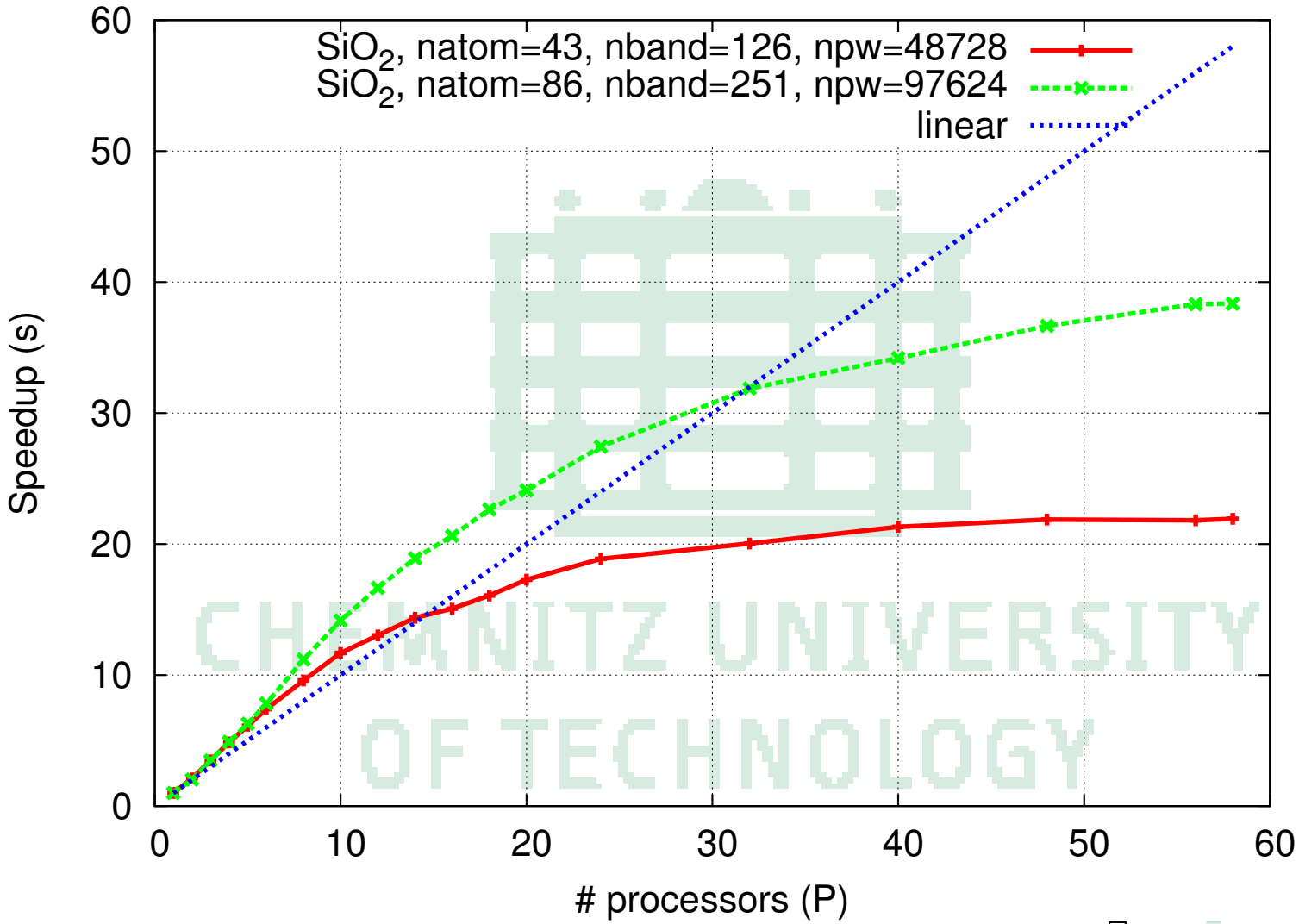
Can we predict parallel Scaling?

- $\Rightarrow$ kind of (comm. overhead as limiting factor)
- ideal scaling: $t(P) = t(1)/P$
  - $\rightarrow \lim_{P\rightarrow\infty} t(P) = 0$
- overhead: $o(P) = o_{red}(P) + o_{a2a}(P)$
  - $\rightarrow \lim_{P\rightarrow\infty} o(P) = \infty$
- crossing point (P_c) denotes maximum scaling
- $t(P_c) = o(P_c)$

Modelled Prediction



Comparison to Benchmarks



Intermediate Conclusions

- Teter's scheme is efficiently parallelizeable
- k-pt, band, and g parallelism can be combined
- parallel scaling can be predicted
- parallel scaling depends on overhead
- overhead depends on system size and LogP parameters

- $\Rightarrow$ overhead is a hard limitation (is it?)
- overlapping could help ;o)

Outline

Non blocking Communication

- Communication can be overlapped with computation
- Progr. model to support overlapping is too complex (threads)
- Non blocking comm. does not change progr. model
- Supported by MPI (MPI_ISEND, MPI_IRECV)

Send/Recv is there - Why Collectives?

- Gorlach, '04: "Send-Receive Considered Harmful"
- $\Leftrightarrow$ Dijkstra, '68: "Go To Statement Considered Harmful"

point to point:

```
if ( rank == 0) then
   call MPI_SEND(...)
else
   call MPI_RECV(...)
end if
```

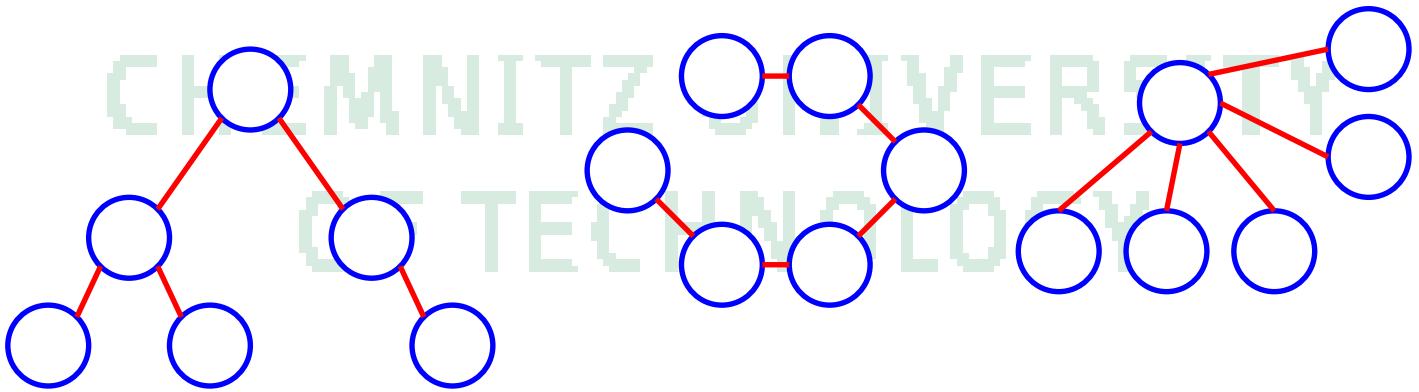
vs. collective:

```
call MPI_GATHER(...)
```

cmp. math libraries vs. loops

Why non blocking Collectives

- overlap communication and computation
- many collectives synchronize unneccessarily
- scale at least with $O(log_2 P)$ sends
- wasted CPU time: $log_2 P \cdot L$
  - Fast Ethernet: L = 50-60 μs
  - Gigabit Ethernet: L = 15-20 μs
  - InfiniBand: L = 2-7 μs
  - $1\mu s \approx$ 4000 FLOPs on a 2GHz Machine



Final Conclusions and Future Work

Conclusions

- Teter's minimization scales ok
- communication overhead is the limiting factor
- parallel scaling is predictable (not easily)
- scaling could be enhanced with overlapping communication and computation to hide latency
- collective communications should be preferred
- $\Rightarrow$ non-blocking collective operations
- LibNBC `http://www.unixer.de/NBC`

Future Work

- use non-blocking collectives to enhance QM codes
- e.g., overlapping schemes for 3D-FFT

The Teter Algorithm

- Steepest descent: $\vec{d}^i = -\frac{\partial f}{\partial \vec{x}^i} = -G\vec{x}^i$
- $f(\vec{x}) \rightarrow E$ Kohn Sham Energy Functional
- $\vec{x} \rightarrow \psi_e$ Wave function for each electron
- $G \rightarrow H$ Hamilton Operator
- Teter's scheme:
  1. 1: check residual for convergence
  2. compute steepest descent vector
  3. orthogonalize it to all bands
  4. compute preconditioned steepest descent
  5. orthogonalize it to all bands
  6. compute conjugate gradient vector
  7. step into cg direction
  8. goto 1

Verifying the Predictions

- Kielmann's `logp-mpi` benchmark:
  $L = 9.78\mu s, o = 0.05\mu s, g = 0.01\mu s$