# Accelerating Ab Initio Nuclear Physics Calculations with GPUs and Optimization Techniques

Masha Sosonkina[1]

[1]Old Dominion University and Ames Laboratory

NTSE'13, Ames IA, 05/13–17/2013

# Collaborators and Acknowledgments


- *D. Oryspayev, H. Potter*, P. Maris, J.P. Vary, – ISU.
- S. Binder, A. Calci, J. Langhammer, R. Roth – TU Darmstadt.
- E. Saule, Ü. Çatalyürek – OSU.
- A. Shirokov – Moscow State University.
- L.T. Watson – Virgina Tech.
- S. Wild – ANL.

## Funding



# Outline

- Ever larger nuclei and bases spaces are required for accuracy. Thus, calculations need to be scaled further.
  - NCSM calculations (e.g., in MFDn) are memory-bound.
  - Compute power is unbounded thanks to multicore and *accelerators*[1] – save for power consumption and computer-size increase.
- Calculations powered by heterogeneous programming models:
  - Examples: OpenMP–MPI [2009, 2012], GPU [2012], 2015 ?
  - Emergence of new architectures (e.g., Titan @ ORNL).

[1] Caveat: lead to deep heterogeneity

# GPU hardware organization

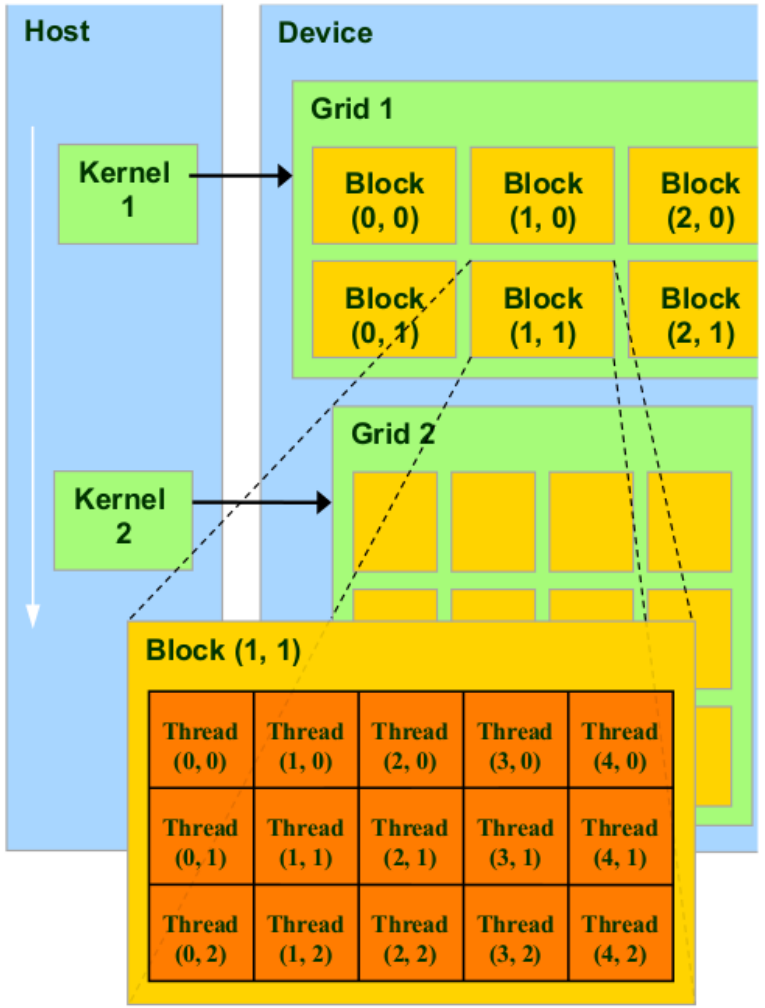
- Significant difference in CPU and GPU hardware.
- GPUs are optimized for throughput rather than latency.
- GPUs – on-chip memory management;
  CPU – cache hierarchy.

*The GPU Devotes More Transistors to Data Processing*


- GPU has faster maximum processing rate.
- CPU$\longleftrightarrow$GPU transfer times is a significant portion of the overall computation.

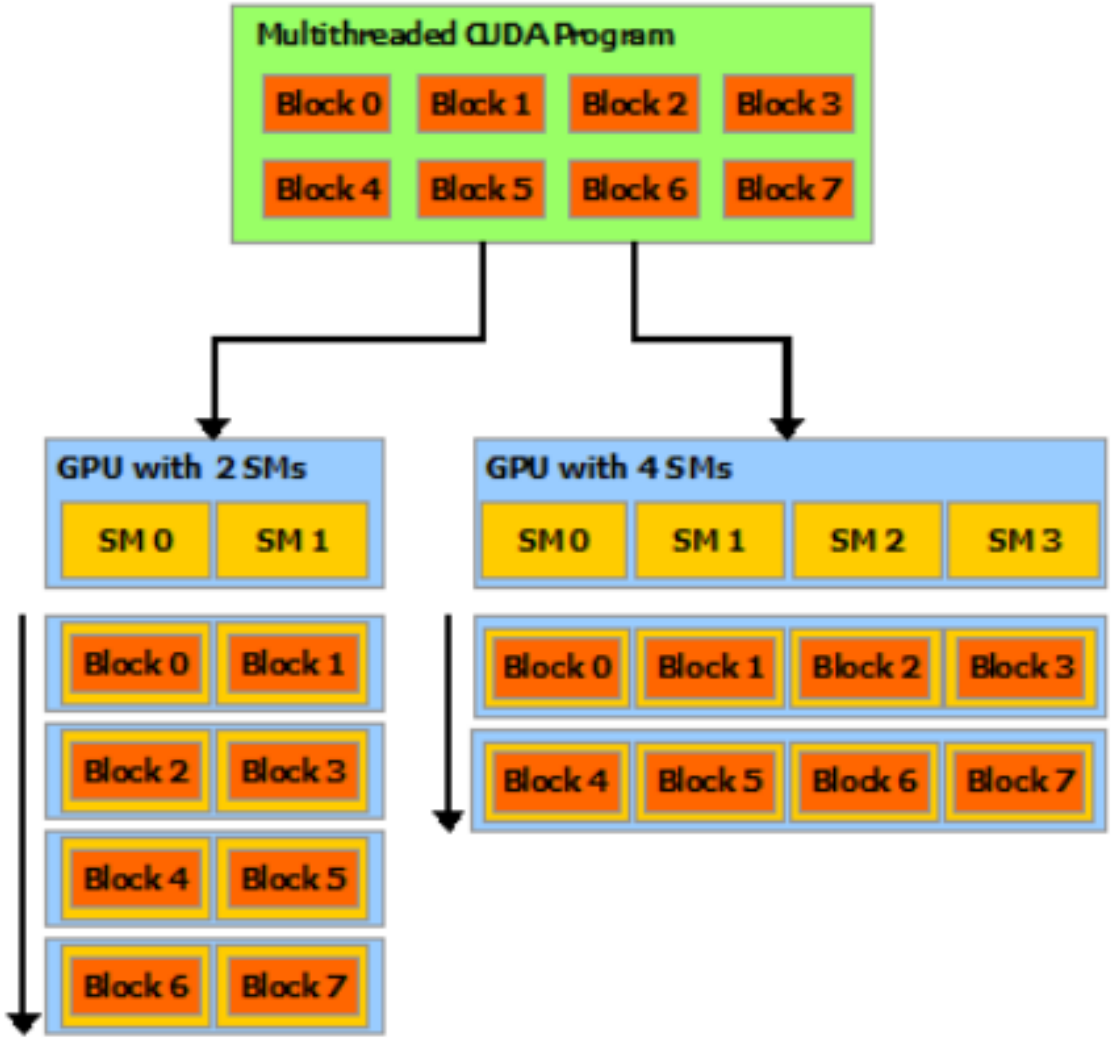
# CUDA programming model

- GPU has its own memory as DRAM and runs many threads in parallel.
- GPU threads are extremely lightweight with small creation overhead.
- Data-parallel application parts are executed on the device as *kernels*, which are many-threaded.

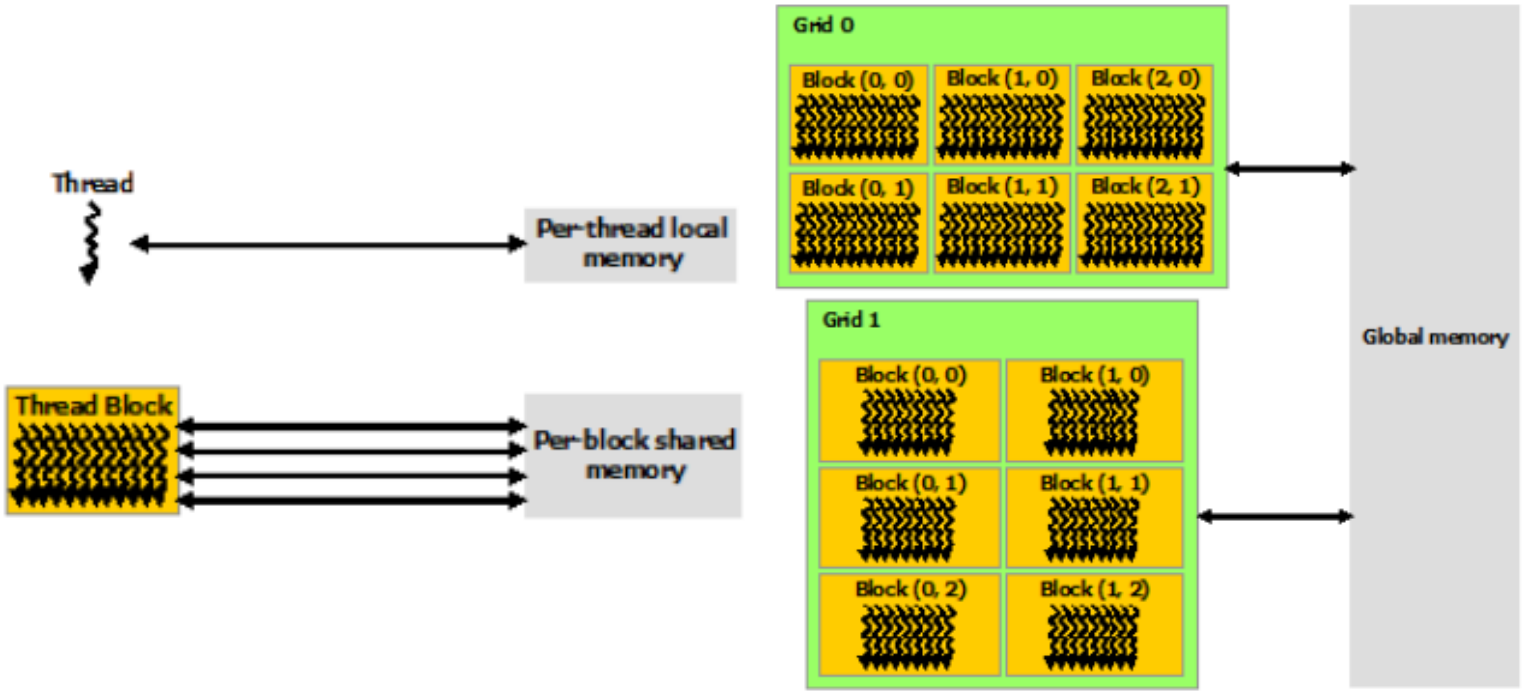


# CUDA Programming model II

- Independent scheduling of blocks facilitates throughput.



# CUDA Programming model – memory layout



# Heterogeneous parallelism opportunities in NCSM

- Hamiltonian diagonalization is the most compute-expensive part.
  - It has been optimized with OpenMP–MPI [ISU collaboration with LBNL].
  - It is difficult to put on GPUs.

| | | | | |
|---|---|---|---|---|
| 1 | | | 12 | 14 |
| 2 | 4 | | | 15 |
| 3 | 5 | 7 | | |
| | 6 | 8 | 10 | |
| | | 9 | 11 | 13 |

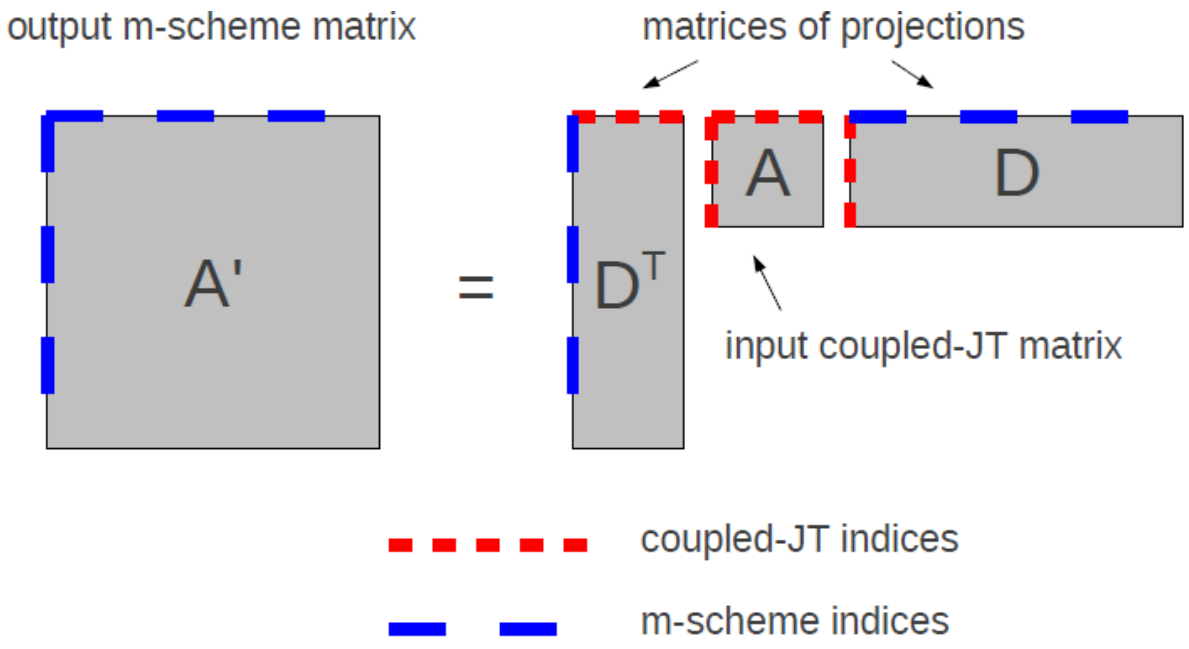
**Hamiltonian matrix calculation may be parallelized.**

- Elements are computed independently.
- Elements are constructed in part from a (possibly large) 3-body interaction matrix.

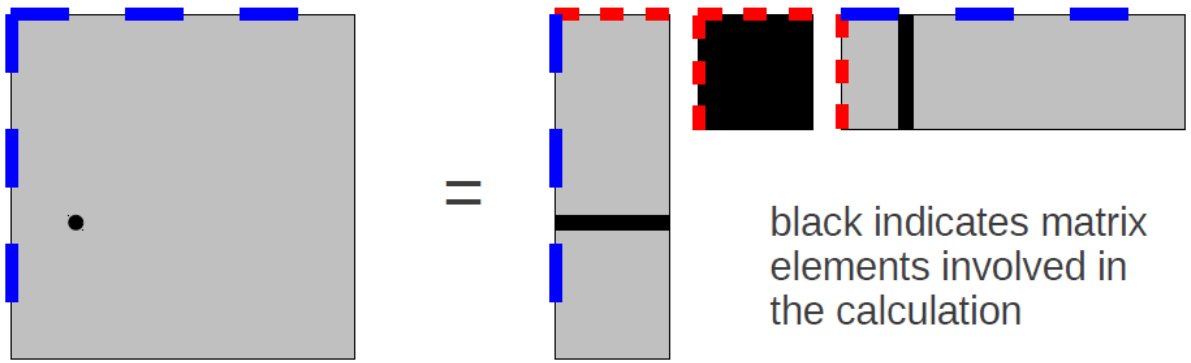
# Matrix elements calculation for MFDn

- MFDn uses ***m***-scheme basis.
  - convenient to use but memory-intensive
- new procedure (ME3M) has been developed to reduce the memory footprint [Roth group at TU Darmstadt].
  - stores 3-body interactions in a more compressed basis by coupling isospin and angular momenta.
  - compressed basis is denoted as *coupled-**JT***.
    - Example: a 3-body matrix of 100GB in the ***m***-scheme basis about $\equiv$ to a matrix only of about 2GB in the coupled-***JT***.
  - Basis transformation is performed every time a 3-body ***m***-scheme matrix element is needed.

# Basis transformations

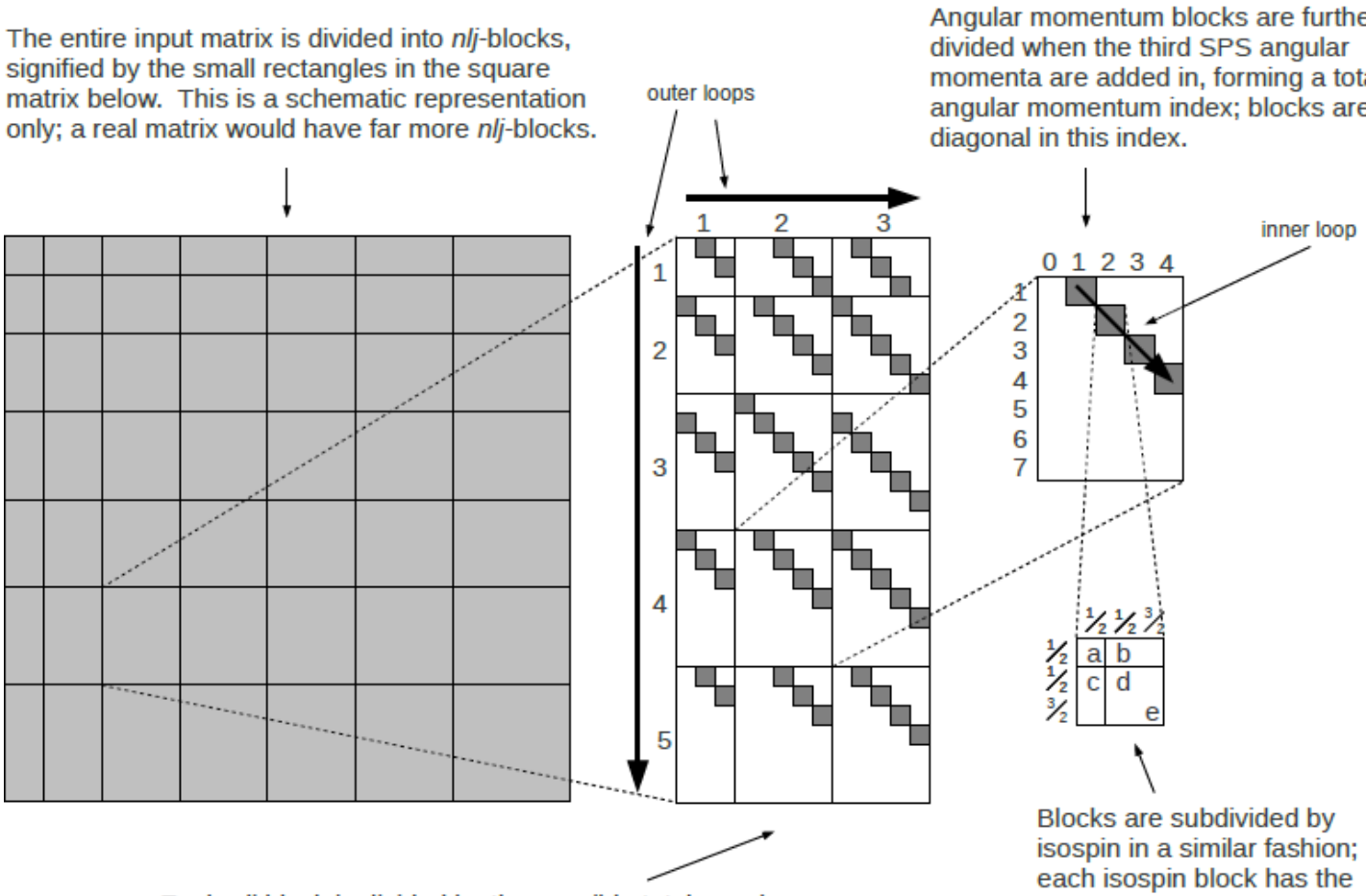


# Close-up for one element construction




- several ***JT***-basis elements are involved in one ***m***-scheme element construction.
- these ***JT***-basis elements are cleverly grouped for efficient access.

# Sparsity structure of the coupled-*JT* matrix



# Basis transformation algorithm

**Algorithm 1** To transform coupled-$JT$ to $m$-scheme

```
L1 ← determineFirstLoopBound
L2 ← determineSecondLoopBound
for 1 to L1 do
  for 1 to L2 do
    L3 ← determineInnerLoopBound
    for 1 to L3 do
      add input matrix elements weighted by CGs
    end for
    accumulate inner loop sum weighted by a CG
  end for
  accumulate second loop sum weighted by a CG
end for
return accumulated sum
```

# GPU implementation

- CUDA was used for generating the ***m***-scheme 3-body matrix elements.
- Each element is calculated by its own CUDA thread.
- One CUDA kernel is invoked over many elements at once.
- GPU-CPU transfer
  - chunk of matrix elements is requested at once.
  - chunk request has sets of single particle state (SPS) indices.
  - group of calculated 3-body ***m***-scheme matrix elements is retrieved from GPU.

# Experimental results

## Platform:

- GPU-equipped node cluster Dirac at NERSC.
- Each node has two Intel 5530 2.4 GHz QPI quad-core Nehalem processors.
- NVIDIA Tesla C2050 “Fermi” with 3GB memory and 448 parallel CUDA cores.
- CPU tests were using 8 OpenMP threads.

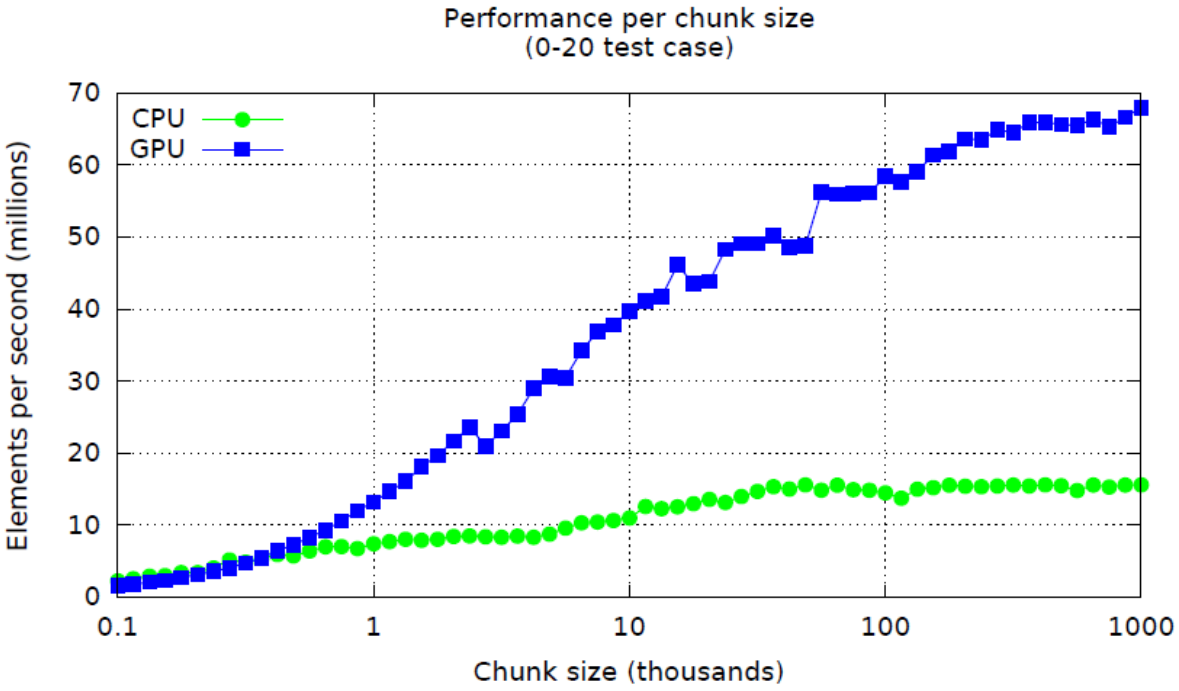
## Test sizes:

- Depending on the desired accuracy of the MFDn calculation, 3-body matrices of different sizes are used.
- Tests are “standalone”, without integration into MFDn.

# Test descriptions

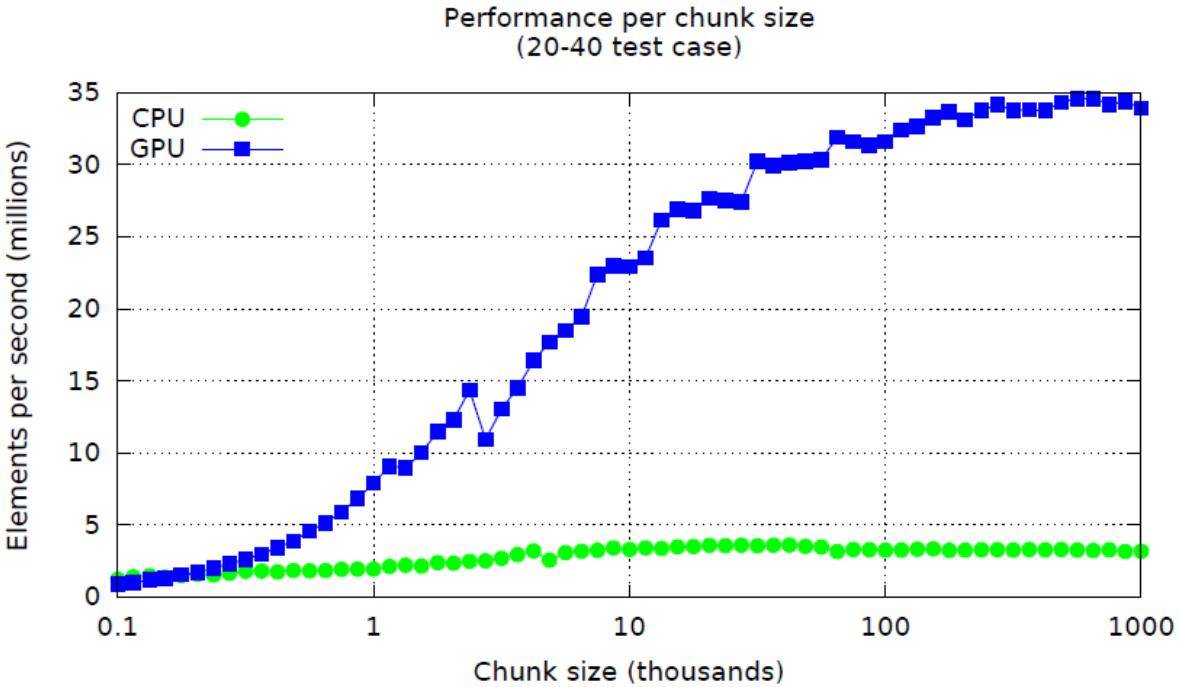
- The largest test cases employ 3-body matrices with 456,000 elements.
- Test I: SPS index range [0–20]
  - short decoupling loops $\rightarrow$ small amount of computation.
- Test II: SPS index range [20–40]
  - more compute-intensive but no increase in memory transfers.

# Experimental results: Performance comparisons



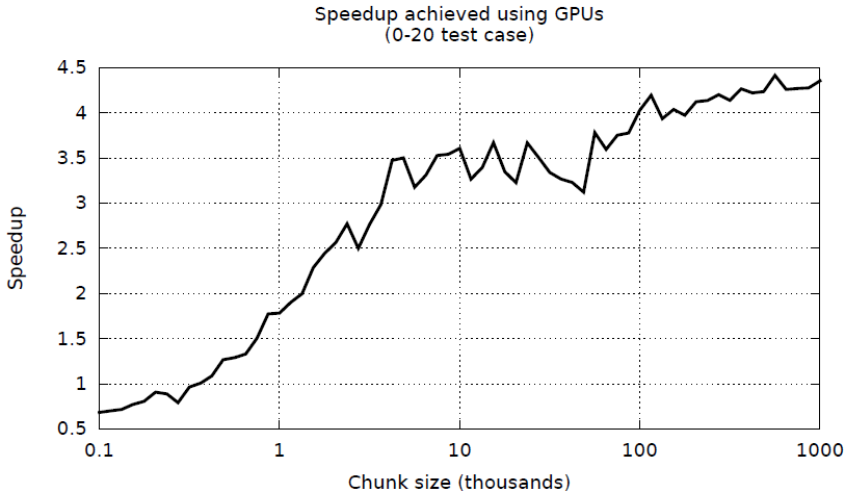

- GPU $\longleftrightarrow$ CPU transfers are included in timings but not one-time memory (de)allocations.

# Experimental results: Performance comparisons

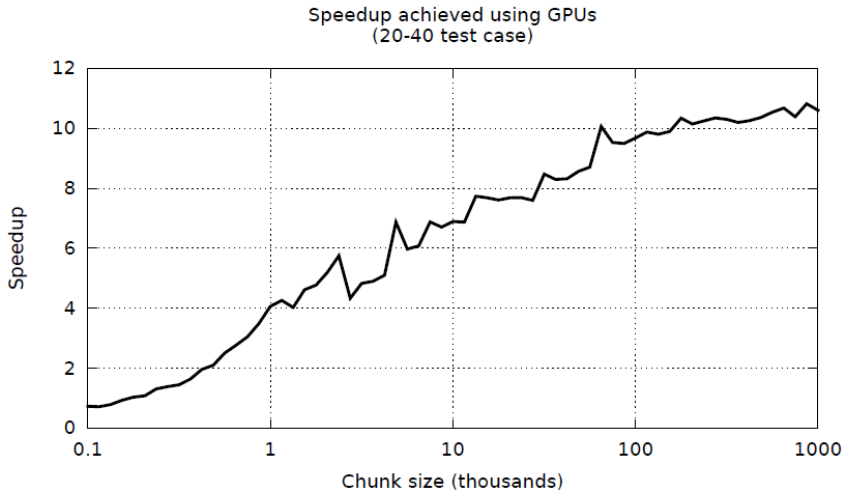



- Results confirm that GPU favors more computation per thread.

# Experimental results: Speedup




Speedup of 4.4




Speedup of 10.8

- Better GPU utilization is desirable ← Now GPU is used at 2.5% and 6.5% of its peak for Test I and Test II, respectively.

# Optimization without derivatives

## Applications in nuclear physics:

- Many tasks in *ab initio* nuclear structure calculations may rely on search techniques:
  - Construction of compressed interaction Hamiltonians,
  - Use of realistic basis functions,
  - Application of external fields.
  - ...
- Due to the size and volume of the nuclear structure calculations, these searches need to be automated.
  - Up to 20 parameters may be searched.

Parallel optimization techniques reduce the search time.

# POUNDerS

Practical optimization using no derivatives (for squares)

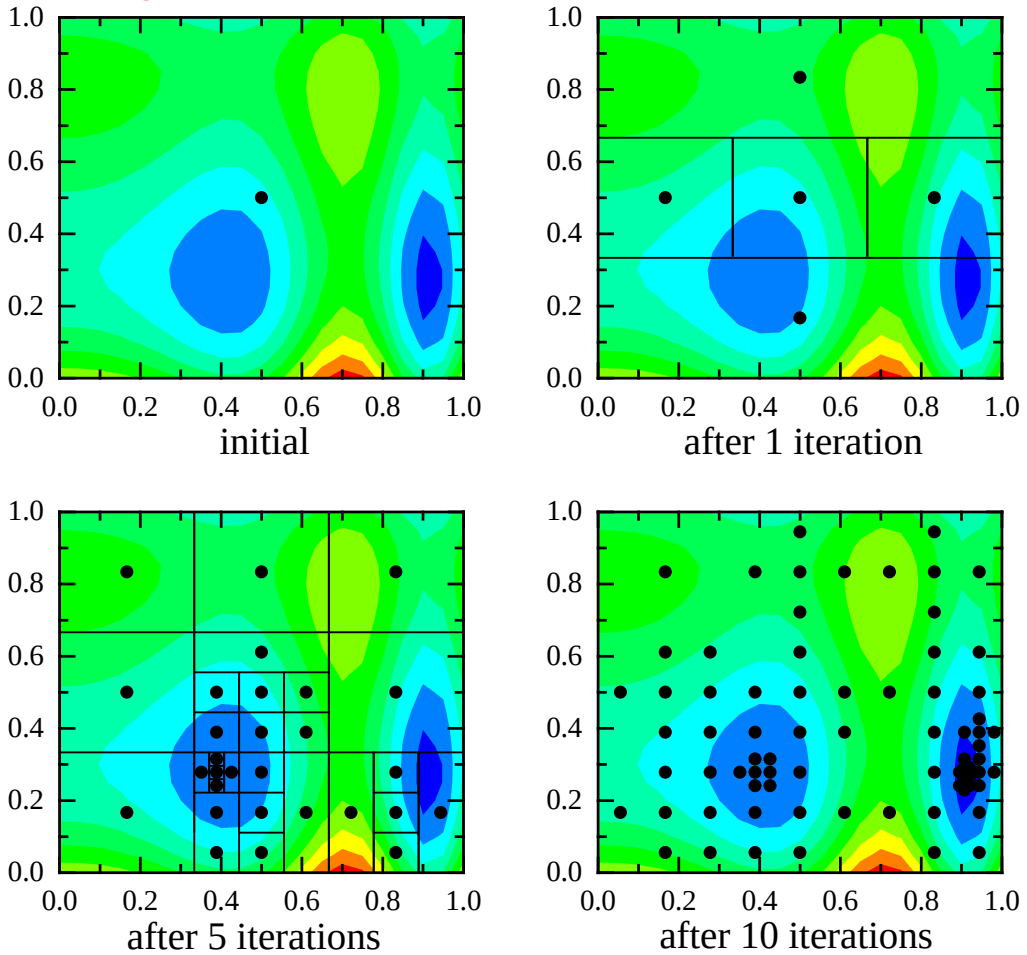
## Algorithmic steps

- Used to minimize the weighted sum of squared errors ($\chi^2$).
- Formulates quadratic model for each searched parameter by approximating the first and second derivatives within a certain trust region.
- Proceeds with Newton-like iterations while growing and shrinking the trust region as needed.

- POUNDerS has been successfully used in estimating Skyrme of energy density. **[ M. Kortelainen, T. Lesinski, J. Moré, W. Nazarewicz, J. Sarich, N. Schunck, M.V. Stoitsovm and S. Wild, *Nuclear energy density optimization*, Phys. Rev. C 82 024313 (2010) ]**
- Preliminary results by A. Shirokov.

# DIRECT Global Search Algorithm: Overview

DIviding-RECTangles, [Jones et al., 1993]



# Outline of DIRECT algorithm

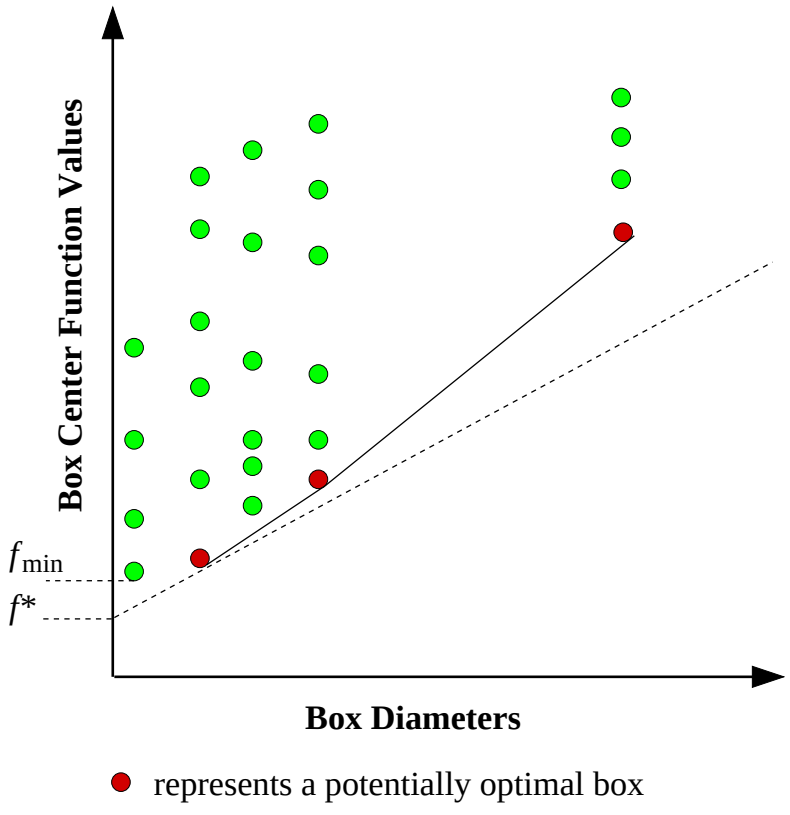
DIRECT finds the global minimum of a function $\boldsymbol{f(x)}$ inside an $\boldsymbol{N}$-dimensional box $\ell \leq \boldsymbol{x} \leq \boldsymbol{u}$. Each iteration of DIRECT consists of the following three main steps.

## Algorithmic steps:

1. **SELECTION:** identifies a set $\boldsymbol{S}$ of "potentially optimal" boxes with dimension $\boldsymbol{N}$ that are subregions inside the design domain.
2. **SAMPLING:** evaluates new points around the center of each "potentially optimal" box in $\boldsymbol{S}$.
3. **DIVISION:** subdivides "potentially optimal" boxes in $\boldsymbol{S}$ based on the function values at the newly sampled points.

Observations: single start point and data dependency.

# DIRECT: Global convergence property




## Box selection rule

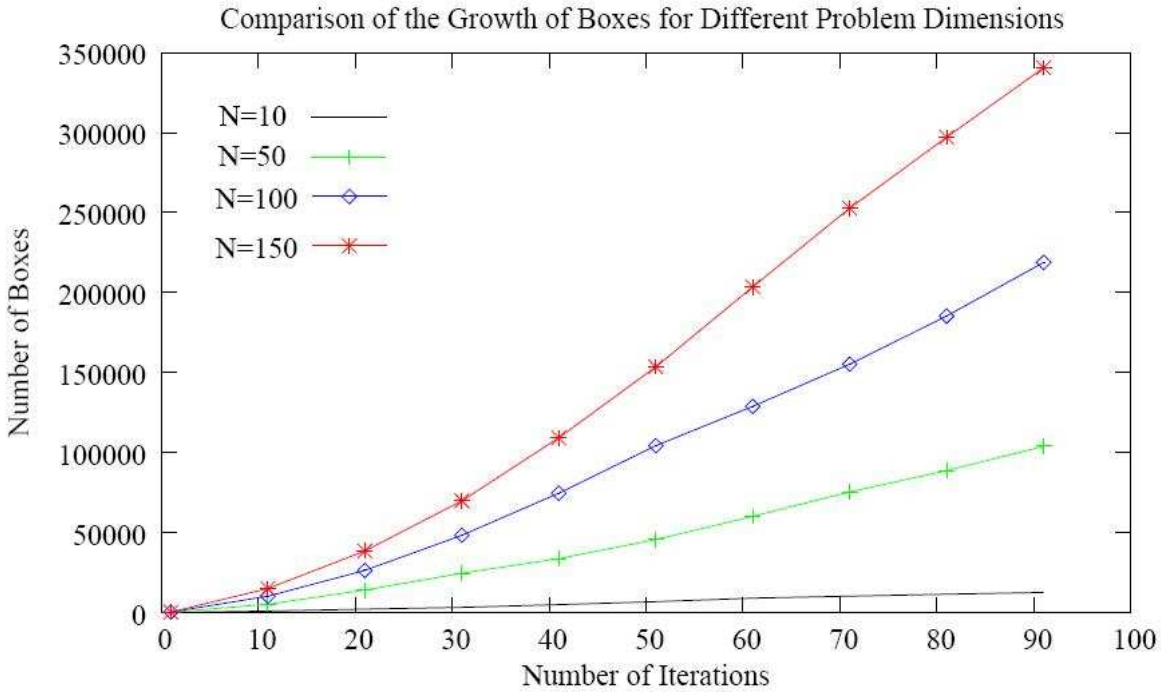
- Box $j$ is potentially optimal if

$$f(c_j) - \tilde{K} d_j \leq f(c_i) - \tilde{K} d_i,$$
$$f(c_j) - \tilde{K} d_j \leq f_{\min} - \epsilon |f_{\min}|,$$

for some $\tilde{K} > 0$ and $i = 1, \ldots, m$;
$m$ is the total number of subdivided boxes

Lipschitz continuity is required in the domain

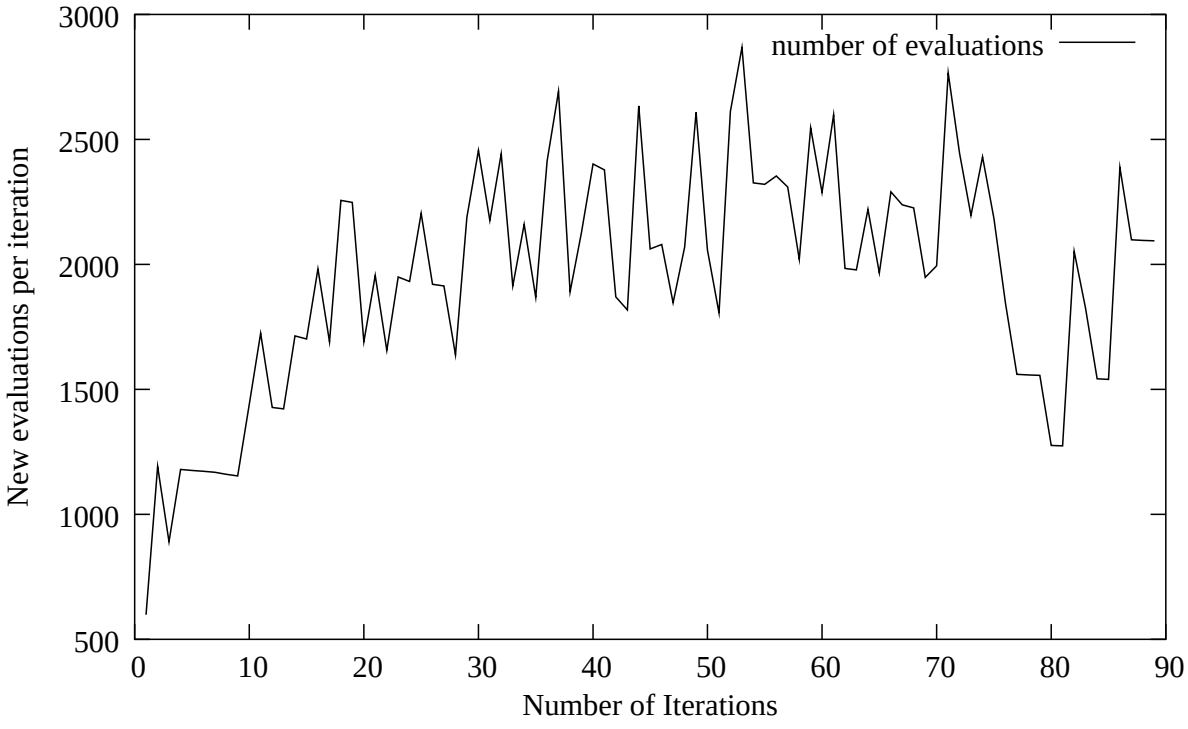
# Varying memory requirements




Objective function:

$$f = 1 + \sum_{i=1}^{N} x_i^2/500 - \prod_{i=1}^{N} \cos(x_i/\sqrt{i}))$$

# Unpredictable workload




Objective function:

$$f = \sum_{i=1}^{N} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$$

# Theoretical and experimental nuclear energy matches

## Example of optimization needs using MFDn

- Computed spectra is too compressed vs experimental.
- Can the same *modified* Hamiltonian, used for the isotopes of the nuclei with atomic mass of 48 ($^{48}$***Ca***), describe other heavy nuclei?
  - Match theoretical and experimental energy levels to find these parameters.
  - Use $\chi^2$ as matching criterion.

# The $\chi^2$ function

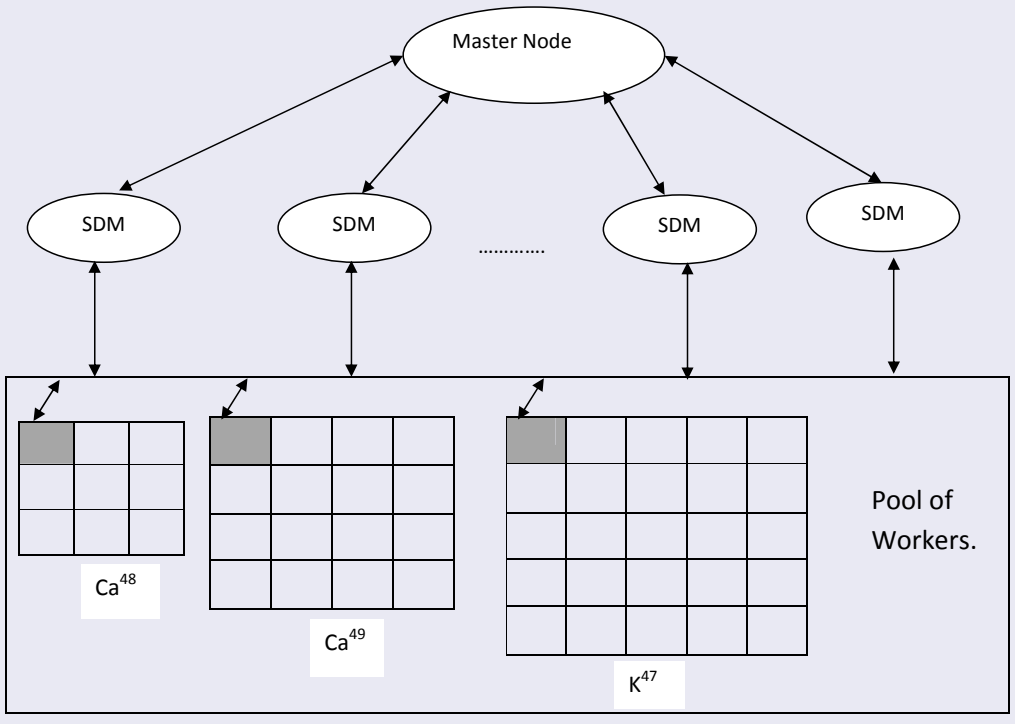
- Important to choose a good objective function to express the best match between experimental and theoretical values.
- $\chi^2$ function serves as the objective function for the search algorithm. Definition of $\chi^2$ function is as:

$$\chi^2(\mathbf{v}) = \sum_{\substack{1 \le i_t \le 15 \\ 1 \le i_e \le k}} \left[ E_{i_e}(\mathbf{v}) - \tilde{E}_{i_t}(\mathbf{v}) \right]^2 \times \sigma_{i_e}^2, \qquad (1)$$

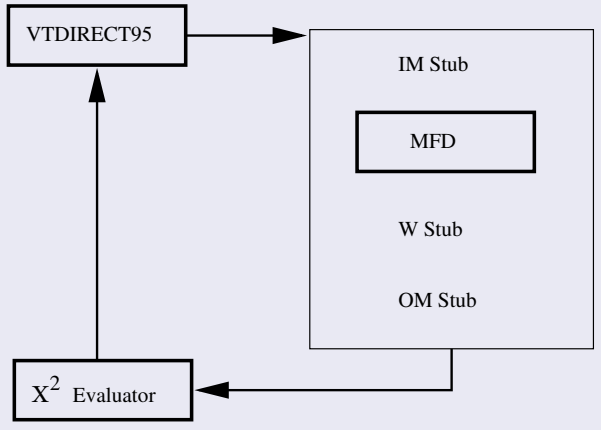
  where $\mathbf{v} = (V_0, V_1, V_{tens})$; $E$ and $\tilde{E}$ are the absolute experimental and theoretical energies, respectively; $i_e$ and $i_t$ are the indices of the corresponding *matched* energy levels (one $i_e$ paired with one $i_t$).
- Each experimental energy level $\ell_e$ is assigned the weight $\sigma_{\ell_e}$. This weight is inversely proportional to the distance of that energy level from the ground energy level.

# Hierarchical design

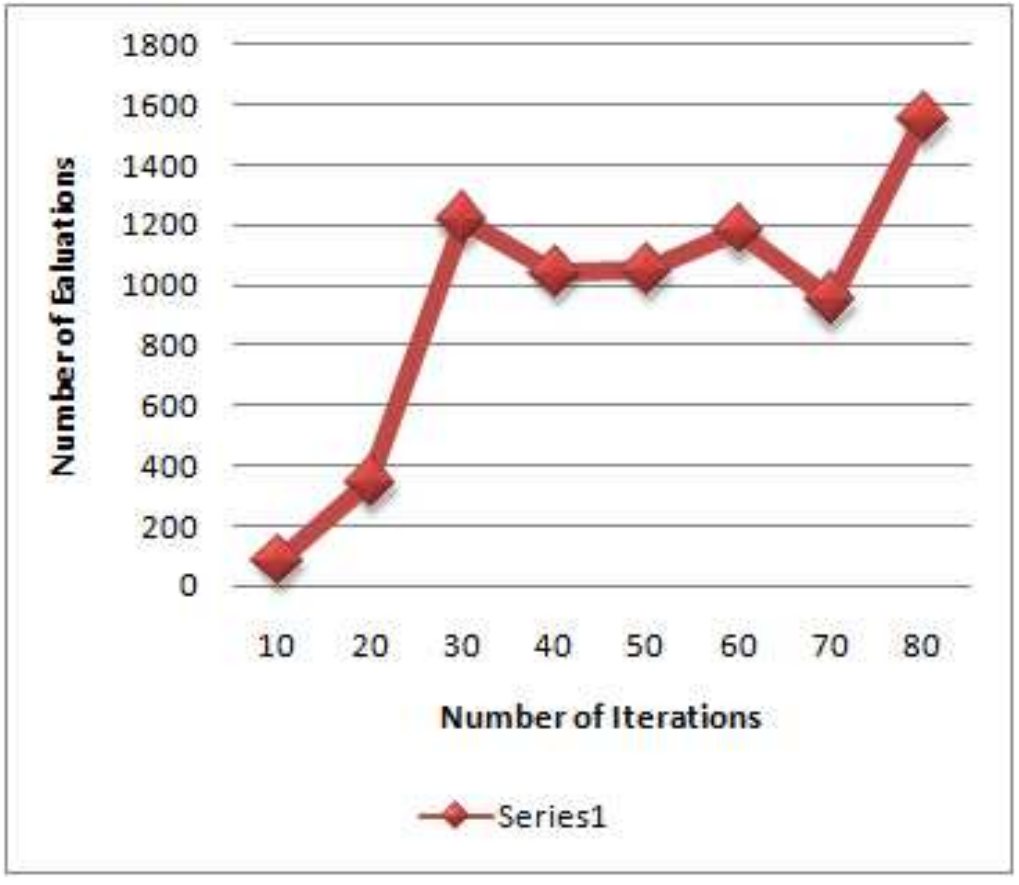
## Spacial view




## Workflow view



# Number of MFDn calls




## Computational setup:

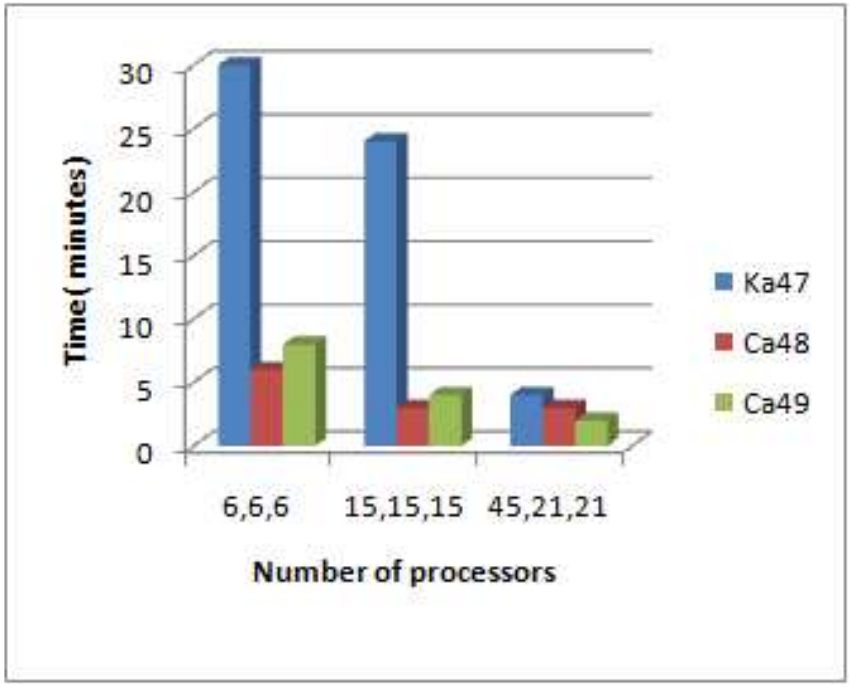
- $^{48}Ca$.
- In VTDIRECT, favor local minimum with $\epsilon = 0$.

## Observation:

Number of function evaluations *per one VTDIRECT iteration* has an increasing tendency with a total number of iterations.

# Execution time for different nuclei

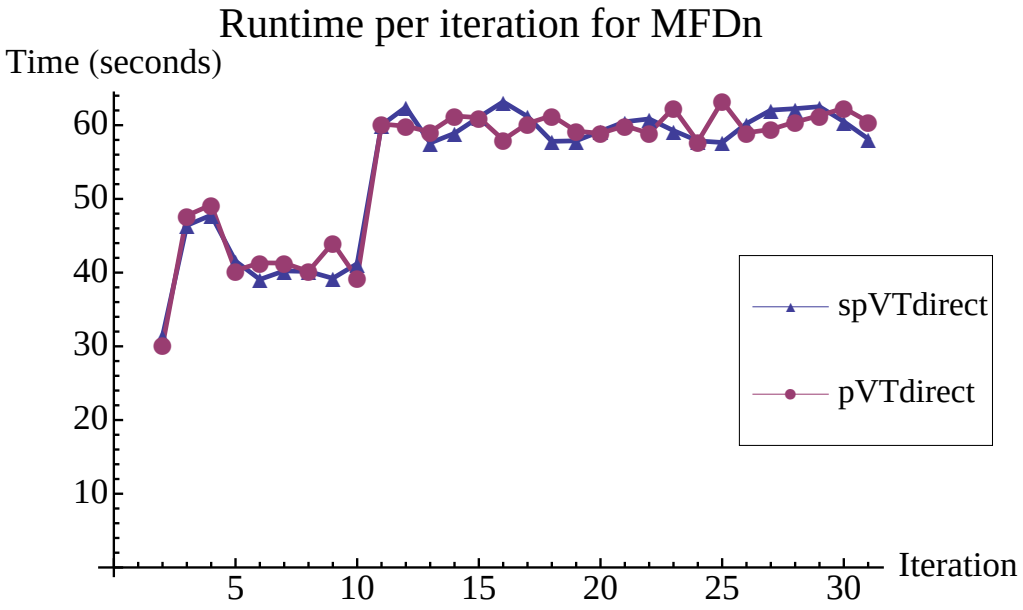
- Multiple nuclei are combined in a single VTDIRECT run $\rightarrow$
  Multiple function evaluations by MFDn within one optimization iteration.
- Different nuclei take different amount of time to execute when equal processor numbers are used.
- Change the number of processors to get approximately the same execution time for each nucleus.




## Computational Setup:

- Nuclei: $^{47}K$, $^{48}Ca$ and $^{49}Ca$ with
- Hamiltonian sizes: 136231, 12000, and 15666.

# MFDn with spawning masters



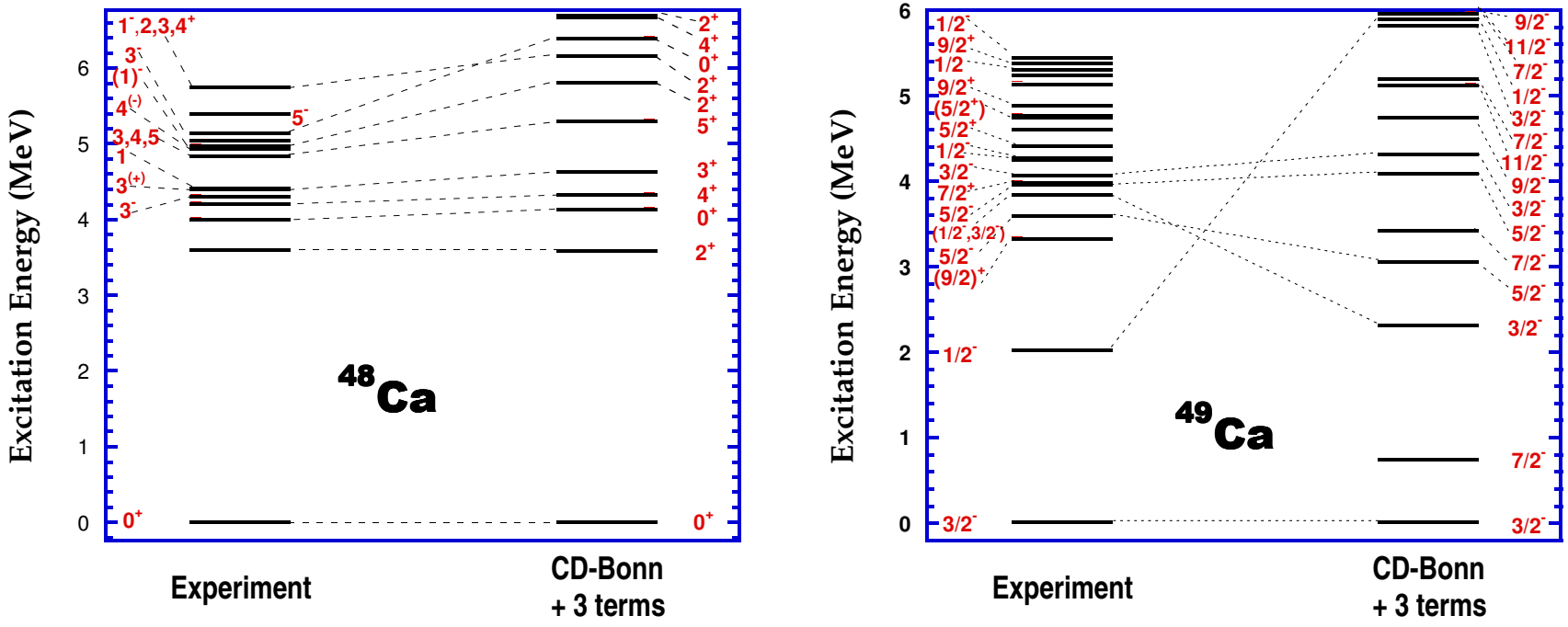

## Computational setup:

- Two MFDn executions per iteration $\sim$ **15**s each.
- Each evaluation is spawned on several processes.

## Observation:

New master spawning overhead is overshadowed by high-cost function evaluations (which are spawned anyway).

# Results




(a) $^{48}Ca$ (b) $^{49}Ca$

Matching of experimental and theoretical energy levels.

# Summary and future work

- Obtained promising speedups using GPUs in construction of the Hamiltonian matrix.
- Experimented with derivative-free search algorithms.

## Future work

- Incorporate the GPU implementation into MFDn (work in progress).
- Investigate other opportunities of using GPUs in MFDn.
- Apply efficiently derivative-free search algorithms to a diverse set of calculations.