Integral and tensor interfaces and their use on supercomputers

Johann Pototschnig

Laboratoire de Chimie et Physique Quantique, Toulouse, France

Sorbonne Université, Paris, 24.10.2022

Challenges of modern computing infrastructure

- heterogeneity
  - processing units (CPUs, Accelerators, GPUs, TPUs)
  - memory spaces (shared disks, RAM, CACHE L1/L2/L3)
  - connections (InfiniBand, ...) (speeds vary for memory spaces)

→ compilation/optimization every time

- task distribution
  - avoiding idle processing units
  - avoid communication bottlenecks

→ optimized task management (static / on the fly)

- portability (software/performance)
  - supercomputers have a limited lifetime (about 3 years)
  - software gets regularly updated

→ use standards

4608 nodes
2 CPUs per node
6 NVIDIA VOLTA GPUs
500 GB RAM
picture from https://commons.wikimedia.org/wiki/File:Summit_(supercomputer).jpg
node design: `https://docs.olcf.ornl.gov/systems/summit_user_guide.html#summit-nodes`

Challenge in relativistic computations - memory

Table: Number of basis functions / orbitals for different systems

element	range	basis	nao	nocc	nvir	$nvir^4$
SF_6	-5 .. 100	DZ	204	24	126	2.5E+8
		TZ	346	24	229	2.8E+9
		QZ	600	24	386	2.2E+10
UF_6	-5000 .. 20	DZ	551	146	197	1.5E+9
La-Porphyrin	-5 .. 10	DZ	1093	67	550	9.1E+10

- memory:
  - 1C, nonrel. : 9.1E+10(orbitals)*8(real, double prec.) $\rightarrow$ 733 GB
  - 2C, relativistic : 1.5E+12(spinor)*16(complex, double prec.) $\rightarrow$ 23425 GB
- problem dependent expansion:
  - augmentation
  - polarization
  - core correlation

Use of libraries

- separate code that provides functionalities using an interface
- advantages
  - highly performant / optimized code due to focused development
  - compilation / optimization by staff at computing facilities
- disadvantages
  - less control about code
  - optimization for different requirements
- requirement
  - standardized interface (compatibility with different versions)
  - usage by several programs / large user base
  - ongoing development / support
- examples (widespread use)
  - BLAS (vector/ matrix operations)
  - LAPACK (linear algebra functionalities)
  - OpenMP/MKL (shared memory distributed computation)
  - MPI (distributed memory distributed computation)

Integrals

Integral libraries

focus on GTOs, example for libraries:

Name	algorithm	parallelization	seperate	used by
libint	Head-Gordon–Pople (Obara-Saika)	SIMD, OPENMP	yes	ORCA, MPQC, GAMESS, BAGEL, ORCA, CP2K, PSI4
libcint	Rys	OpenMP	yes	pySCF, Chronus Quantum
GBTOlib	numerical	MPI, OpenMP	yes	UKRmol+, RMT
InteRest	Obara–Saika	OpenMP	no	DIRAC, RESPECT
HERMIT	McMurchie-Davidson	MPI	no	DIRAC, DALTON
AOINTS	Head-Gordon–Pople	OpenMP	no	Q-Chem
ABACUS	McMurchie-Davidson	MPI	no	CFOUR, ACES III, DALTON
MOLECULE	McMurchie-Davidson		no	CFOUR, MOLCAS, MOLPRO
SEWARD	Obara-Saika		no	MOLCAS

codes with their own routines

- MRCC (Obara-Saika)
- NWChem (Obara-Saika-Tracy / McMurchie-Davidson)
- TeraChem (McMurchie-Davidson)

Slater orbitals $\rightarrow$ numerical integration (ADF) plane waves - periodic codes $\rightarrow$ k-space integration

Integral libraries - Interfaces

- libint

```
call compute_eri_f(1, deriv_order, am1, c1, alpha1, A, &
                                   am2, c2, alpha2, B, &
                                   am3, c3, alpha3, C, &
                                   am4, c4, alpha4, D, &
                                   F, erieval)
```

- libcint

```
cint2e_ip1_cart(buf2e, shls, atm, natm, bas, nbas, env, 0_8)
```

- GBTOlib

```
call eri_shell(lena,xa,ya,za,anorms,la,aexps,acoefs, &
               lenb,xb,yb,zb,bnorms,lb,bexps,bcoefs, &
               lenc,xc,yc,zc,cnorms,lc,cexps,ccoefs, &
               lend,xd,yd,zd,dnorms,ld,dexps,dcoefs,
               two_el_column,sph_ints)
```

- InteRest

```
call interest_eri('llll',fijkl,gout,&
                  lk,ek,xk,yk,zk,ck,&
                  ll,el,xl,yl,zl,cl,&
                  li,ei,xi,yi,zi,ci,&
                  lj,ej,xj,yj,zj,cj )
```

Tensors

Selection of tensor libraries

name	dist. mem.	PU	cont.	decomp.	spez. type
Cyclops (CTF)	✓	CPU, GPU	✓	✓	sparse
DISTAL (sp-)	✓	CPU, GPU	✓	x	sparse
ExaTensor	✓	CPU, GPU	✓	x	x
ExaTN	(✓)	CPU, GPU	✓	✓	x
PyTorch	x	CPU, GPU	✓	x	sparse
NumPy	x	CPU	✓	x	x
TACO	x	CPU, GPU	✓	x	sparse
TAL_SH	x	CPU, GPU	✓	x	x
TensorFlow	✓	CPU, GPU, TPU	✓	x	sparse
TiledArray	✓	CPU, GPU	✓	x	block sparse

(PU ... processing units), (cont. ... contraction), (decomp. ... decomposition)
a larger list can be found on arxiv[1]

[1] C. Psarras, L. Karlsson, J. Li, P. Bientinesi; *The landscape of software for tensor computations*, https://doi.org/10.48550/arxiv.2103.13756

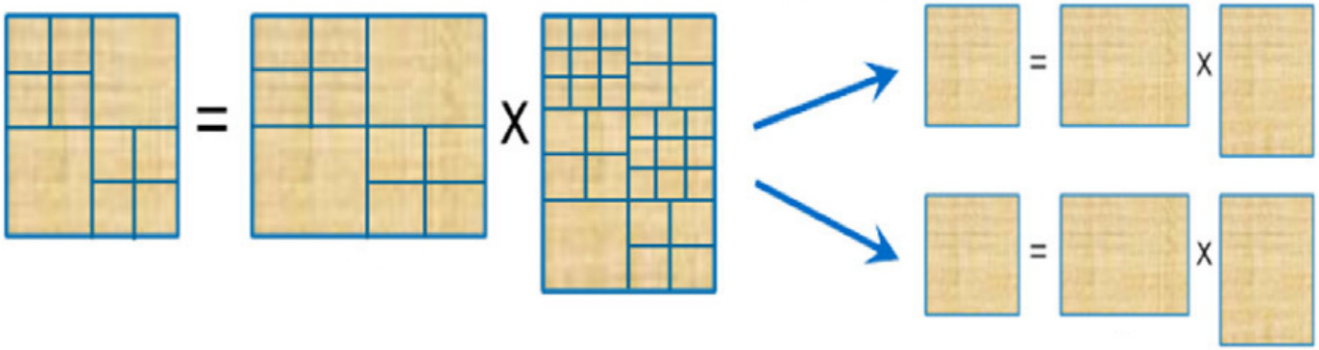
Tensor libraries

- currently there is a change from GPUs to Accelerators, some are developing for TPUs
  $\rightarrow$ heterogeneity will increase and requires adaptation of the codes
- tensor libraries are often designed for a specific purpose
  $\rightarrow$ other functionalities might be inefficient or missing
- task distribution and scheduling is part of the library, but they might become separated (e.g. StarPU, PaRSEC)
- parallelism strategy:
  data / task / mixed based parallelism
  (splitting in domains)
- synchronous and asynchronous communication:
  polling/push communication, one-sided communication

Experiences

ExaCorr Implementation

- a new Coupled Cluster implementation in DIRAC[2] using the ExaTensor tensor library[3]
- ExaTensor
  - tensor library that uses up to several 1000 nodes
  - tensors are distributed over several nodes
  - mainly focused on distributed contractions:



  - node level parallelization by TALSH
- TALSH
  - single node version / part
  - enables the use of GPUs

[2] DIRAC, a relativistic ab initio electronic structure program, written by H. J. Aa. Jensen, R. Bast, A. S. P. Gomes, T. Saue and L. Visscher, see `http://www.diracprogram.org`

[3] Lyakh, D. I., *Int. J. Quant. Chem.*, **2019**, 119, e25926

Parallelization

3 levels of parallelization

- separate memory
    - MPI (MPICH, OpenMPI)
    - required data has to be sent to MPI process
    - parallelization different nodes
- shared memory
    - OpenMP, MKL
    - access the same memory
    - parallelization on a node
- GPUs
    - CUDA

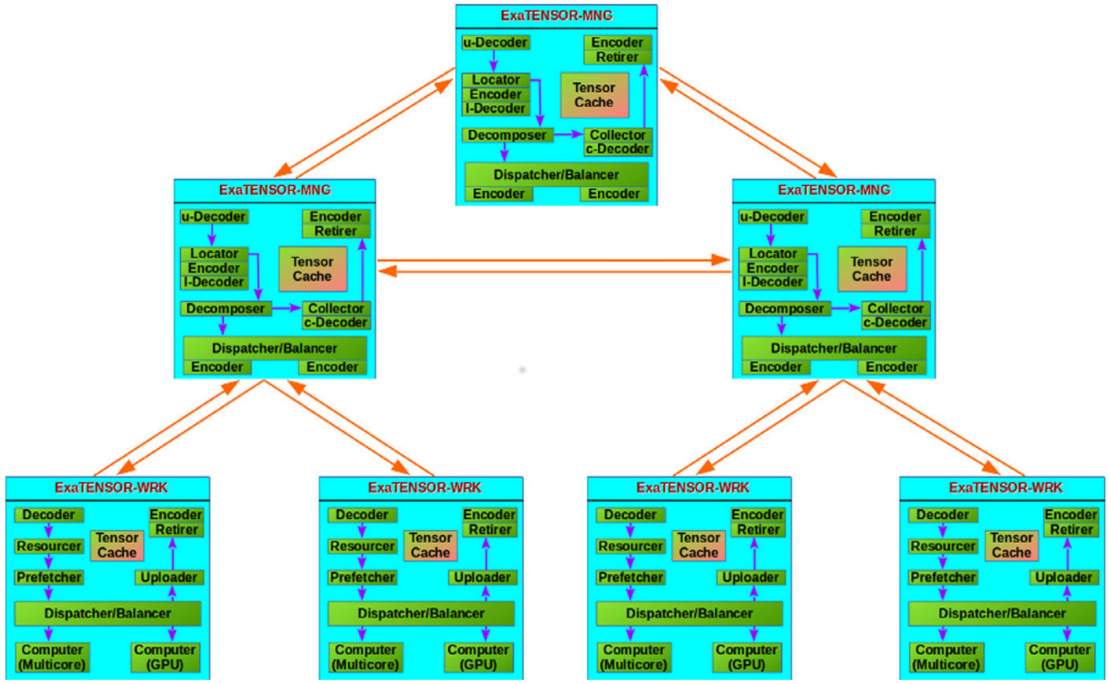


Figure: ExaTensor structure[a]

---

[a] Lyakh, D. I., *Int. J. Quant. Chem.*, **2019**, 119, e25926

ExaTensor code examples

- create tensor

```
ierr=exatns_tensor_create(tensor,"tensor_name",tensor_id,
tensor_root,EXA_DATA_KIND_C8)
```

- initialize tensor

```
ierr=exatns_tensor_init(tensor,'ZERO')
```

- contraction

```
ierr=exatns_tensor_contract(
"S(a,b,i,j)+=V(a,b,c,d)*T(c,d,i,j)",s2,vvvv,t2,scalar)
```

- apply a method

```
ierr=exatns_tensor_transform(tensor,method)
```

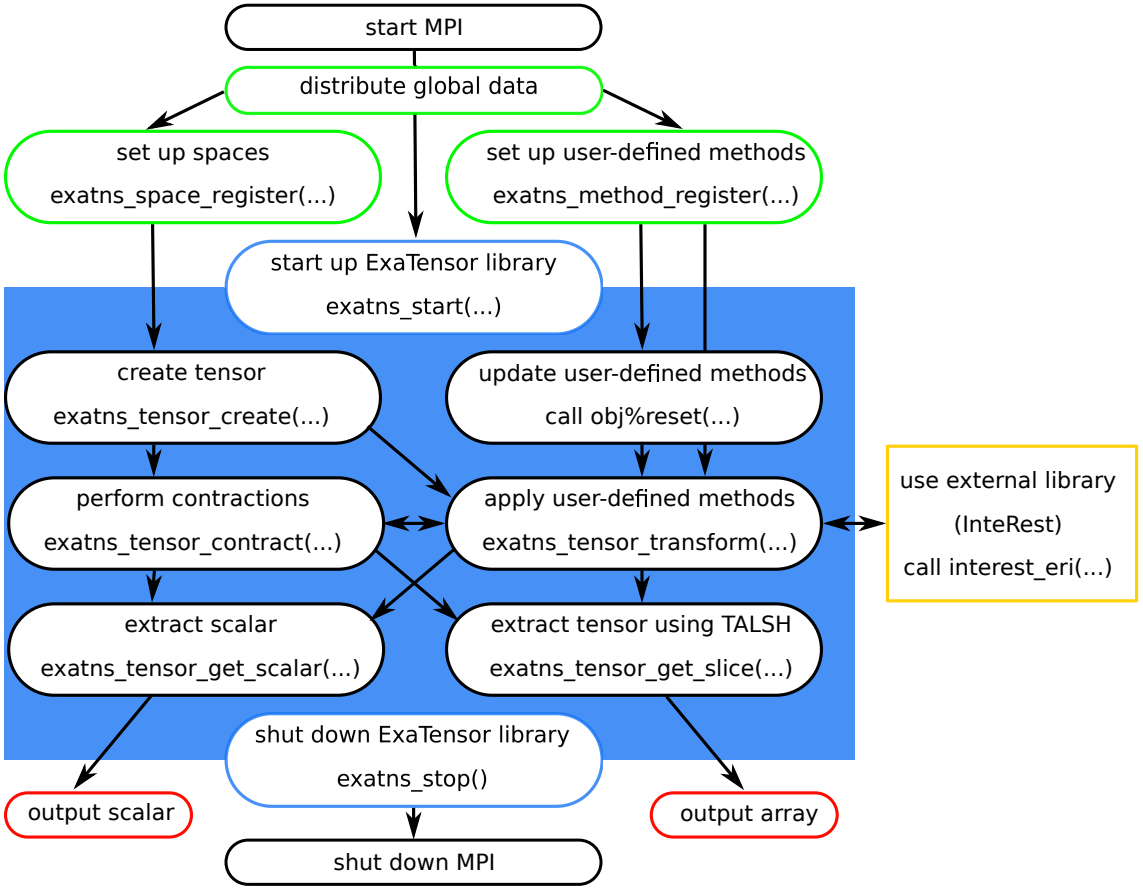
- extract results:

```
ierr=exatns_tensor_get_scalar(tensor,value)
```

- release memory

```
ierr=exatns_tensor_destroy(tensor)
```

Interface[4] to ExaTensor



[4] J. V. Pototschnig, A. Papadopoulos, D. I. Lyakh, M. Repisky, L. Halbert, A. S, P. Gomes, H. J. Aa Jensen, L. Visscher, *J. Chem. Theory Comput.*, **2021**, 17, 5509-5529, doi: 10.1021/acs.jctc.1c00260

Experiences - Implementation

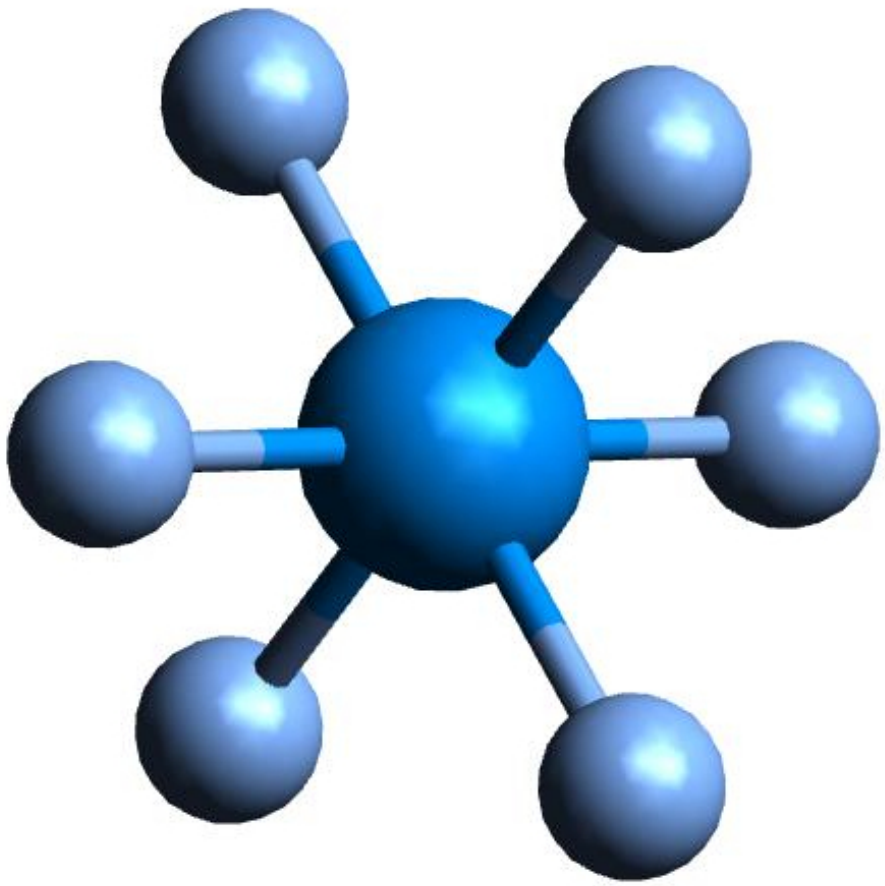
- interface allows quick implementation
  (especially if efficiency optimizations are hidden)
- sometimes additional functionalities are required
  $\rightarrow$ cooperation with the developer
  $\rightarrow$ access to the source code
  (e.g. update of orbital energies for denominator)
- newer code uses modern functionalities of compiler
  $\rightarrow$ stability / testing of this functionality might be lacking
  (e.g. ExaTensor works with GNU 8, 11 and higher but not with 9 and 10)
- newer code uses modern functionalities of MPI
  $\rightarrow$ functionalities might not be available / tested
  (e.g. routines for one-sided communication (MPI-3 standard) were not available on Fugaku)

ExaCorr - Scaling

- usable up to 2050 MPI processes
- small scaling example:

Table: Timings in seconds for CCSD computations for UF_6 with 146 occupied and 394 virtual spinors.

np	t_I	t_{CC}	t_Λ
32	621	5246	not finished
64	446	4521	not finished
128	462	4138	2712
256	369	3880	2531
512	459	3974	2561



Large run

Table: CCSD expectation value calculations for the uranyl tris-nitrate complex ($[UO_2(NO_3)_3]^-$) using the X2C atomic mean-field Hamiltonian

occ.	vir.	np	cost	cost/np	walltime	T1	q_{zz}
			dyall.v2z				
106	534	160	9.14E+14	1.00	3h34	0.0126	10.02
156	534	160	1.98E+15	2.17	5h33	0.0103	9.70
156	818	360	1.09E+16	5.30	11h21	0.0102	8.48
202	534	400	3.32E+15	1.45	8h02	0.0091	9.73
202	680	2050	8.72E+15	0.75	12h28	0.0090	8.89
202	818	2050	1.83E+16	1.56	18h23	0.0089	8.59
202	896	2050	2.63E+16	2.25	23h04	0.0089	8.54
			dyall.v3z				
106	694	480	2.61E+15	0.95	6h	0.0139	10.29
156	694	480	5.65E+15	2.06	10h24	0.0114	10.04
156	944	2050	1.93E+16	1.65	17h48	0.0104	9.99

→ Problem: the number of nodes is determined by the memory demand

Experiences - Computations

- high memory demand, supercomputers have relatively small memory per PU:
  - Summit: 2 CPUs + 6 GPUs + 512 GB RAM / node
  - Fugaku: 50 CPUs + 32 GB RAM / node
  - Lumi: 128 CPUs + 512 GB RAM / node

  → number of nodes was decided by memory demand

  → scaling hard to show as we cannot go to smaller node numbers
- optimization
  - size of tensor pieces
    large packages more efficient, but less parallelization is possible
  - different contractions require different settings, e.g.:
    full vs. partial contraction
    square vs. rectangular contraction

  → size and distribution of tensor pieces / communication settings
- restrictions on supercomputers
  - time limits: supercomputers have rather strict wall time limits (4 - 24 h) → required restart files for computations that need more time
  - only allow computations that use a large number of nodes
    → scripting can be used to run computations in parallel

Molecular Tricks

Reduction of computational effort / memory demand

- for larger system we are running out of steam
  $\rightarrow$ efficient libraries will not be sufficient
- methods used in molecular computations
  1. symmetry
     point groups, antisymmetry, ...
     of course, it is extensively used for atoms
  2. localization
     applid in:
     $\rightarrow$ multilevel methods (e.g. QM-MM)
     $\rightarrow$ linear scaling methods
  3. reducing basis / orbital spaces
     especially for correlated methods, via cutoffs / thresholds / localization
  4. tensor / matrix representation techniques
     e.g. sparse tensors, ...
  5. combinations of strategies

Restricting orbitals

- energy threshold

Table: Ionization potential in eV of YbF for different numbers of correlated spinors employing the dyall.v2z basis set.

threshold$_{low}$	threshold$_{high}$	nocc	nvir	% occ	% vir	CCSD
-20	2.3	49	89	63	21	4.49
-20	6	49	137	63	32	5.89
-20	150	49	267	63	63	5.89
-20	10000	49	367	63	86	5.89
-3	40	31	213	40	50	5.95
-20	40	49	213	63	50	5.90
-60	40	61	213	78	50	5.90
-400	40	77	213	99	50	5.90
exp						5.91±0.05[5]

- natural orbitals[6]
  select optimized orbitals to treat correlation

[5] Kaledin, L. A.; Heaven, M. C.; Field, R. W., *J. Mol. Spectrosc.*, **1999**,193, 285–292
[6] X. Yuan, L. Visscher, A. S. P. Gomes; *J. Chem. Phys.*, **2022**,156, 224108

Tensor Representations

- tensors in 2 dimensions (matrices) have a unique representation by using SVD (singular value decomposition)
- for higher dimension it is not unique anymore
- take as an example the 4 dimensional two electron integrals (depictions [7]
- tensor decomposition
  - computing the representation
  - example: Cholesky

[7] Benedikt, U.; Auer, H.; Espig, M.; Hackbusch, W.; Auer, A. A.; *Molecular Physics* **2013**, 111, 2398.

Example - Cholesky decomposition

Table: Cholesky decomposition for H_2O in the QZ basis

δ	M	% AO	ΔE (H)	$t_{trans.}$	$t_{dec.}$	$\frac{Chol.}{old}$ %	$\frac{dec.}{trans.}$ %
1.0E-03	760	3.1	2.24E-06	865	27	57.7	3.1
1.0E-04	936	3.8	3.14E-08	1044	40	69.6	3.8
1.0E-05	1138	4.6	9.01E-09	1285	58	85.7	4.5
1.0E-06	1409	5.7	2.49E-09	1563	86	104.3	5.5
1.0E-07	1644	6.7	6.54E-10	1874	116	125.0	6.2
1.0E-08	1903	7.7	7.52E-11	2167	154	144.6	7.1

- so far only used in the AO to MO transformation
- can be applied in SCF
- CCSD and EOM can be performed without ever constructing the 4-index quantities[8]

[8] Epifanovsky, E.; Zuev, D.; Feng, X. T.; Khistyaev, K.; Shao, Y. H.; Krylov, A. I. *J. Chem. Phys.*, **2013**, 139, 134105

Summary

challenges for supercomputers

- memory to CPU ratio
- allocation for shorter highly parallel computations
- software availability / compatibility

challenges for libraries

- demand
  integral and tensor libraries are mostly used by quantum chemists
- interface
  no standard defined

Conclusion

Acknowledgments

- HAMP-vQED team




T. Saue J. Brandejs G. Fabbro M. Salmann R. Benazzouk

- ExaCorr team:
  Andre Gomes, Lucas Visscher, Dmitry Lyakh (ExaTensor), Michal Repisky (Integral evaluation), H. J. Aa. Jensen (SCF paralellization), Stan Papadopoulos, Loic Halbert
- Oak Ridge National Laboratory - access to SUMMIT
- ERC (grant No 101019907)

Discussion Initializers

- standards for integral interface
- standards for tensors interface
- pitfalls of porting software
- strategies for memory reduction