Optimization of a Sparse Hypermatrix Cholesky Factorization

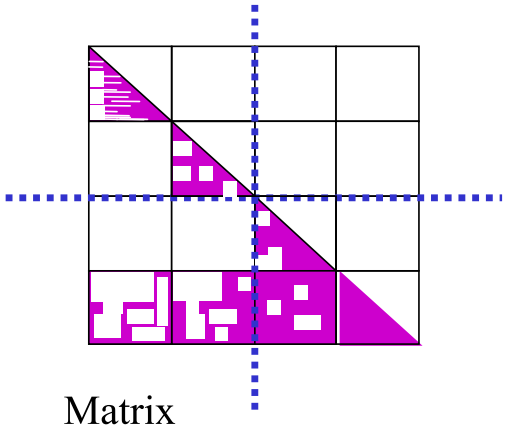
Josep R. Herrero, Juan J. Navarro
{josepr,juanjo}@ac.upc.es

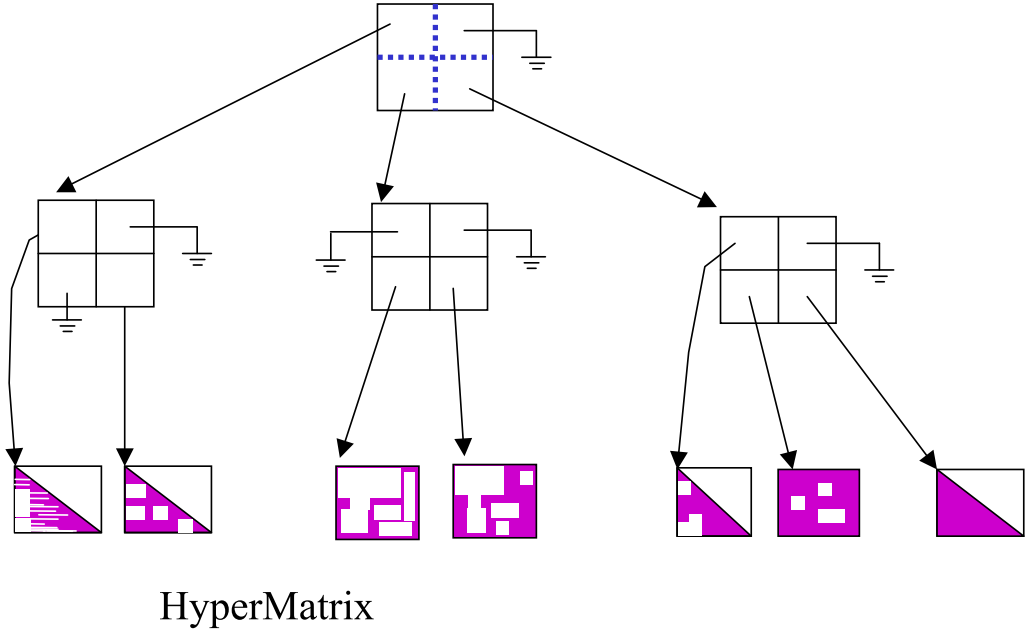
Computer Architecture Department
Universitat Politècnica de Catalunya
Barcelona
Spain

Outline

- Introduction
- Overhead reduction techniques
- Results
- Analysis
- Conclusions

Hypermatrix structure





Overhead

Can store 0's within data submatrices

- Storage
- Computation

Trade-off in data submatrix size

- BLAS3 efficiency
- (Useless) operation on 0's

Reducing Overhead & Increasing Performance

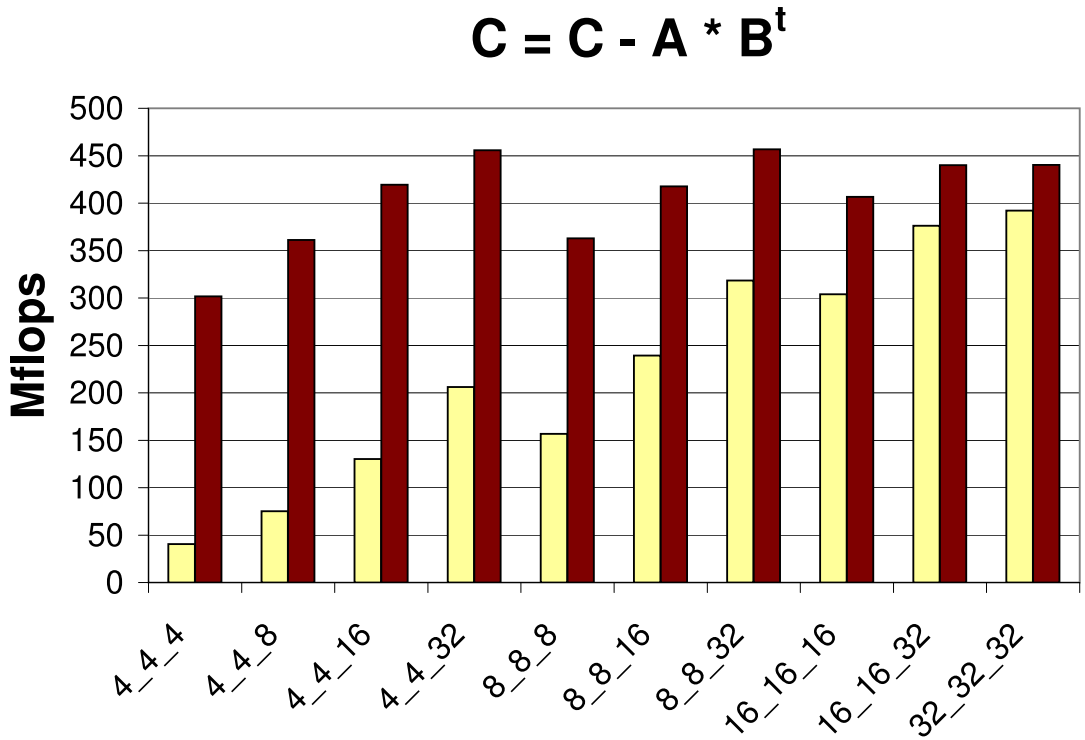
- Efficient kernels which operate on small data submatrices
- Windows within data submatrices

Reducing Overhead & Increasing Performance

Efficient kernels which operate on small data submatrices [Euro-Par'03]

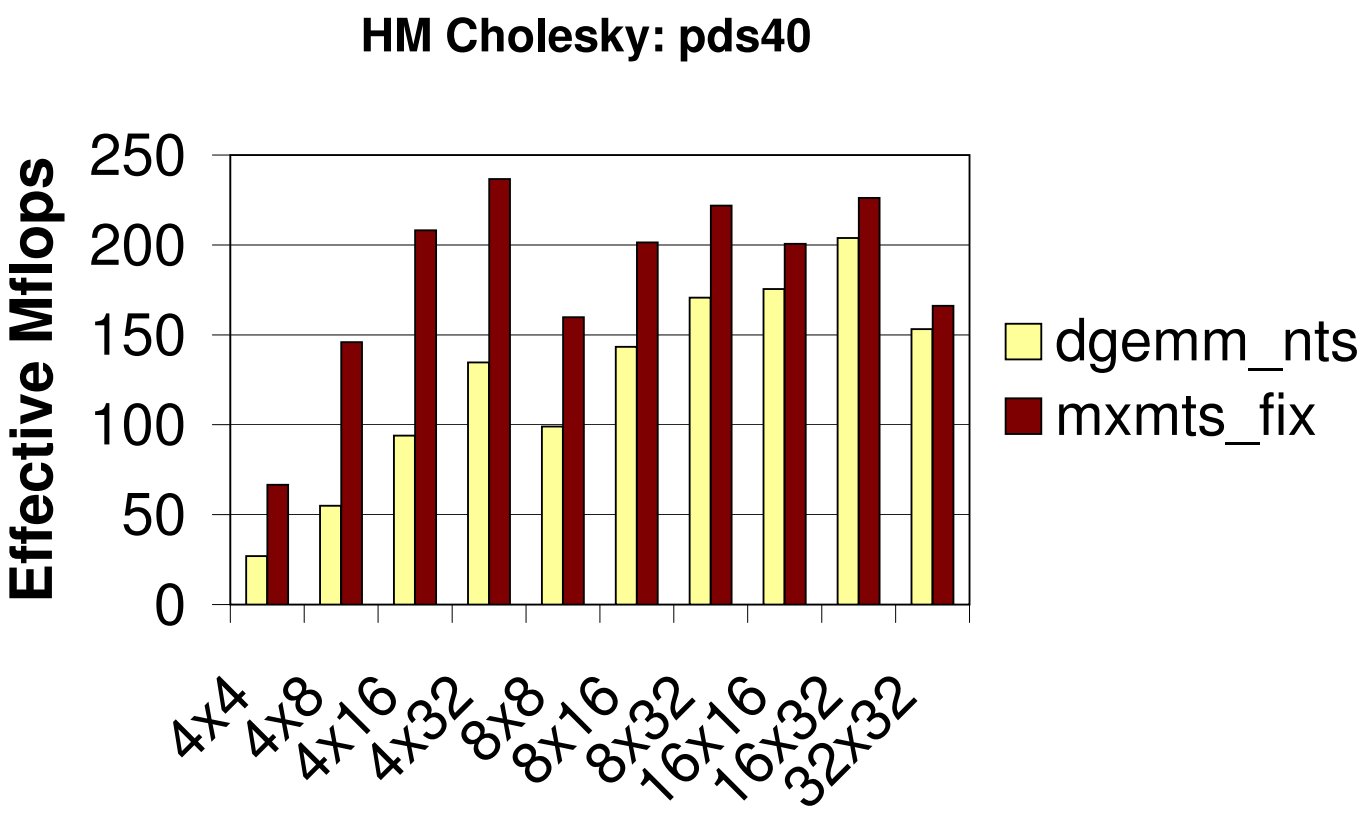
- Fix parameters at compilation time
- Choose best algorithm
  - Loop unrolling factors
  - Loop orders

Matrix multiplication performance on small matrices



R10000 250 MHz (500 Mflop peak)

Hypermatrix Cholesky on problem PDS40



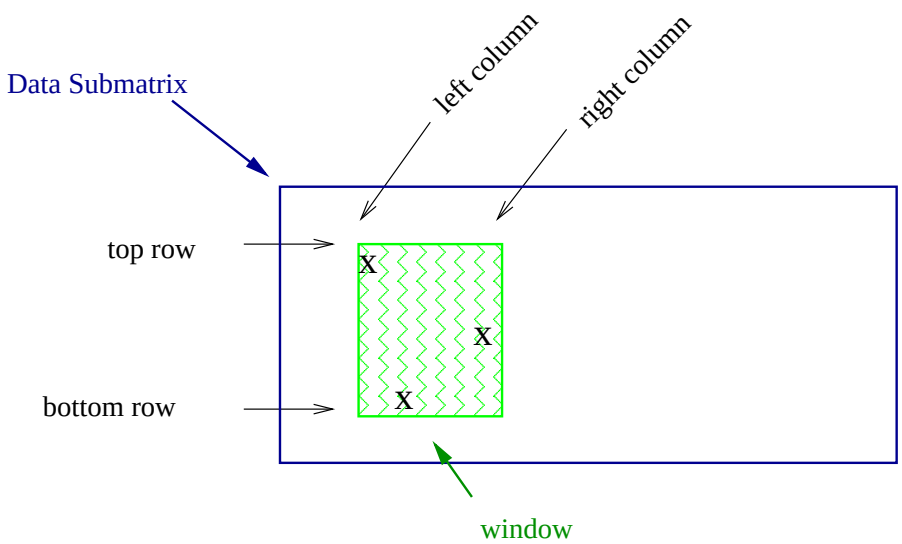
LP problem: Patient Distribution System (40 days)

Reducing Overhead & Increasing Performance

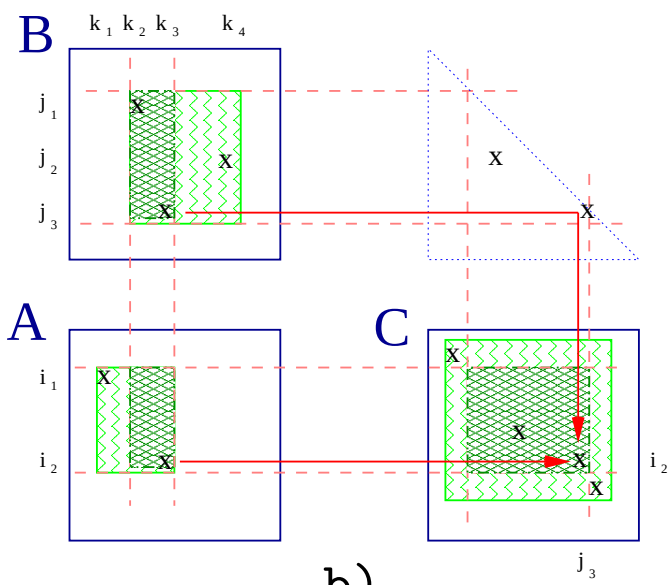
Windows within data submatrices

- Reduce storage
- Reduce computation

Reducing overhead: windows within dense data matrices



a)



b)

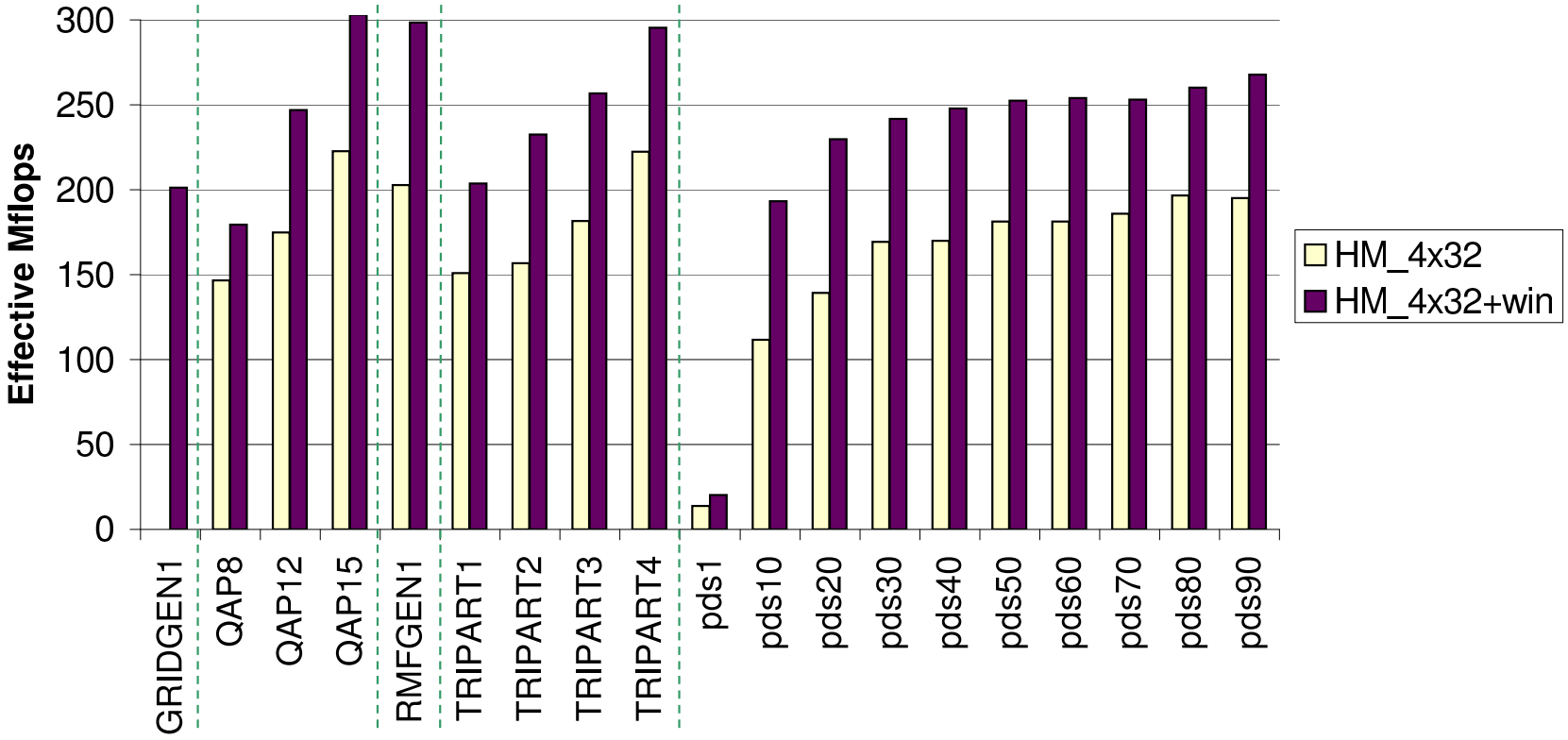
Context information

- MIPS R10000 @ 250 MHz (500 Mflop peak)
- Sequential code
- Data submatrices of fixed size
- Large problems solved In-Core
- Ordered using METIS
- Linear Programming problems
  - NetLib
  - Multicommodity Network Flow generators

Matrix characteristics

Matrix	Dimension	NZs	Factor NZs	Density	Flops in factor
GRIDGEN1	330430	3162757	130586943	0.002	278891960665
QAP8	912	14864	193228	0.463	63764878
QAP12	3192	77784	2091706	0.410	2228094940
QAP15	6330	192405	8755465	0.436	20454297601
RMFGEN1	28077	151557	6469394	0.016	6323333044
TRIPART1	4238	80846	1147857	0.127	511884159
TRIPART2	19781	400229	5917820	0.030	2926231696
TRIPART3	38881	973881	17806642	0.023	14058214618
TRIPART4	56869	2407504	76805463	0.047	187168204525
pds1	1561	12165	37339	0.030	1850179
pds10	18612	148038	3384640	0.019	2519913926
pds20	38726	319041	10739539	0.014	13128744819
pds30	57193	463732	18216426	0.011	26262856180
pds40	76771	629851	27672127	0.009	43807548523
pds50	95936	791087	36321636	0.007	61180807800
pds60	115312	956906	46377926	0.006	81447389930
pds70	133326	1100254	54795729	0.006	100023696013
pds80	149558	1216223	64148298	0.005	125002360050
pds90	164944	1320298	70140993	0.005	138765323993

HM performance with and without windows



Reducing Overhead & Increasing Performance

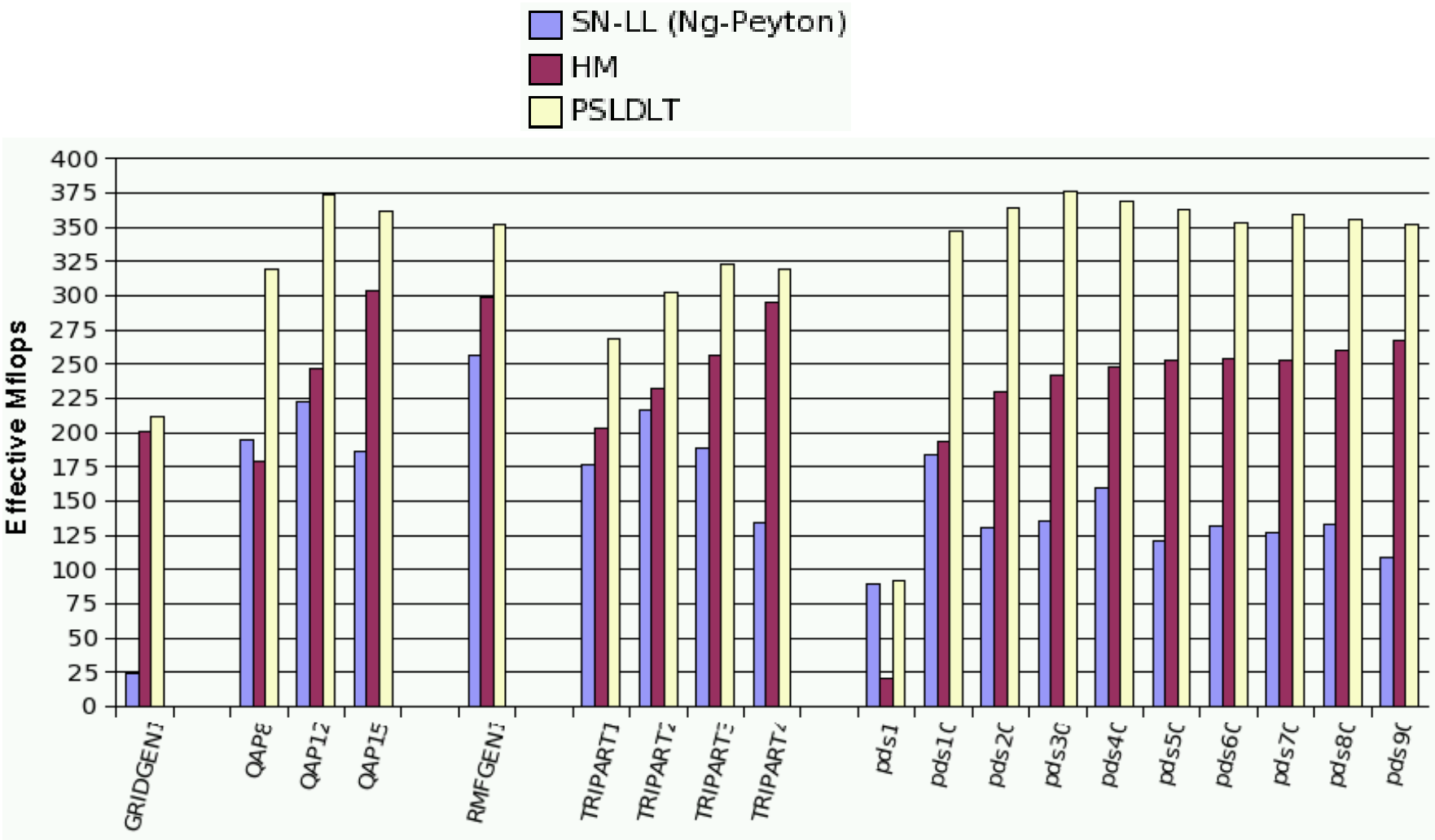
Techniques used:

- Efficient kernels which operate on small data submatrices
  - Rectangular data submatrices
- Windows within data submatrices

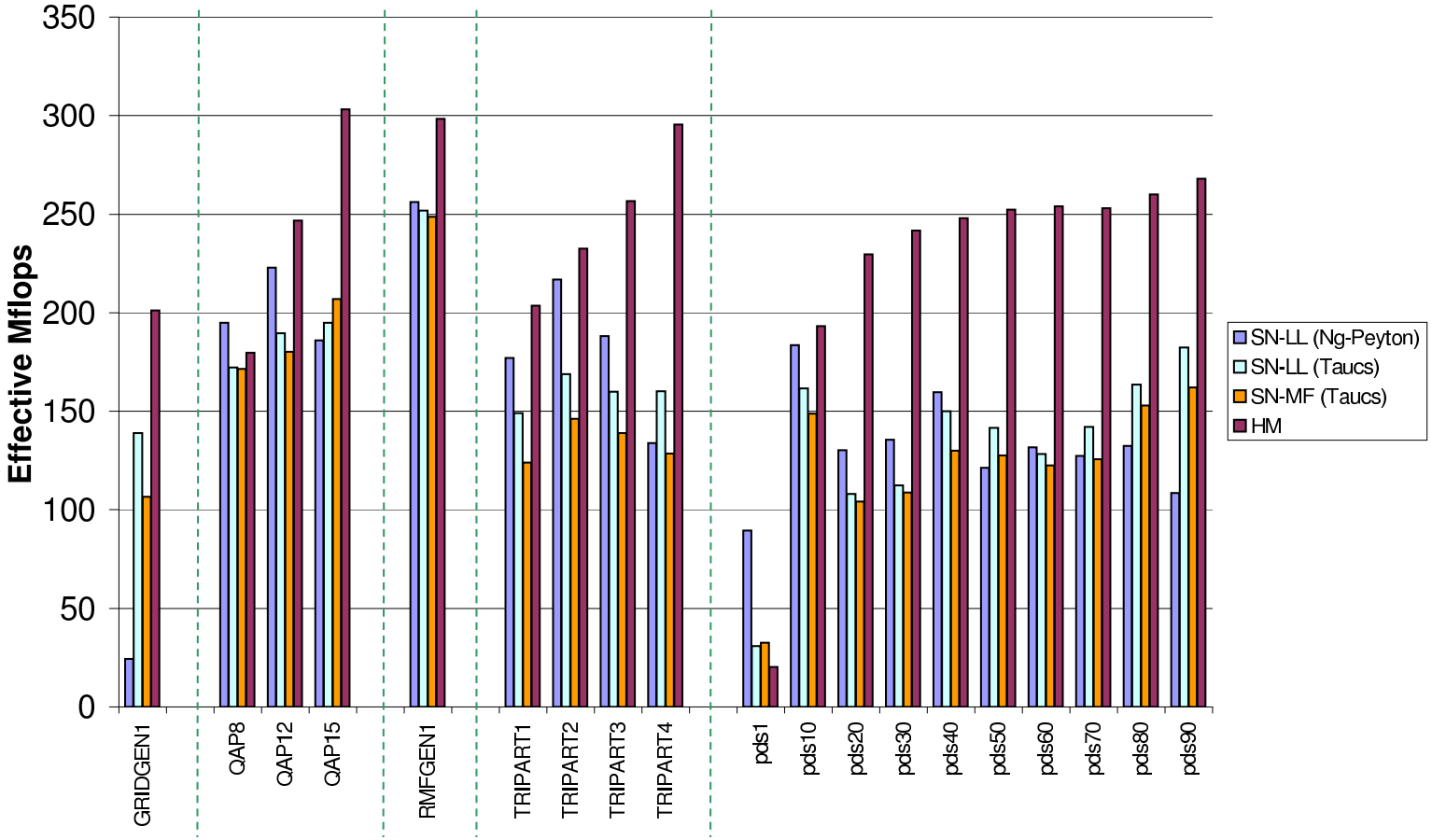
Results:

- Improvement in Hypermatrix Cholesky

Performance: HM vs Ng-Peyton vs PSLDLT



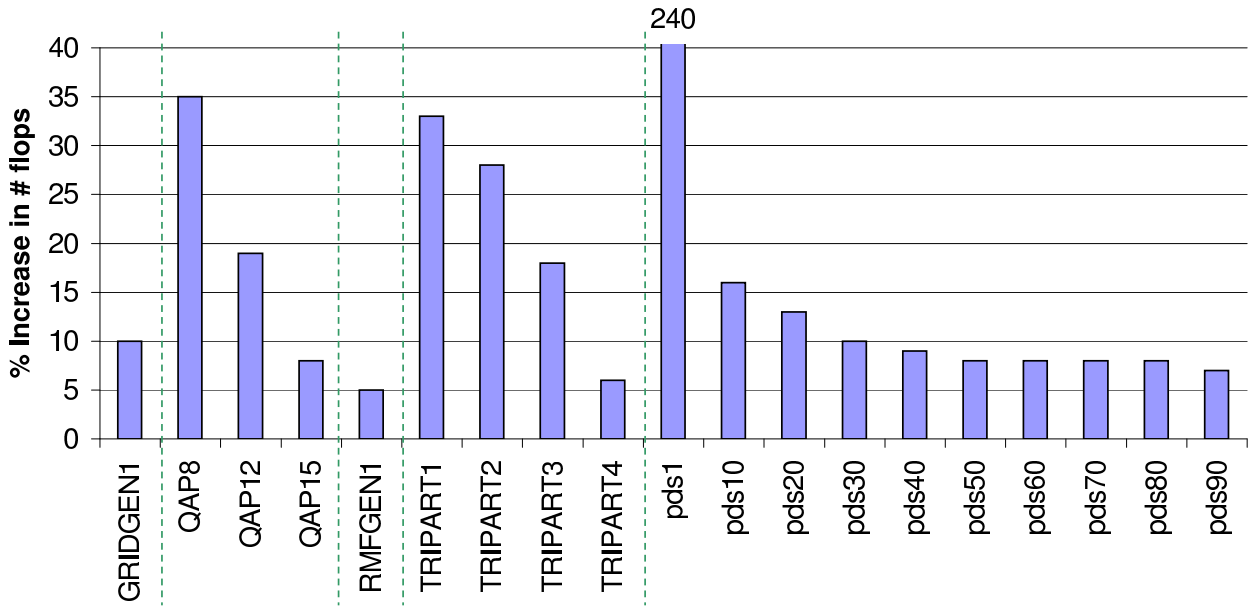
Performance of several sparse Cholesky factorization codes.



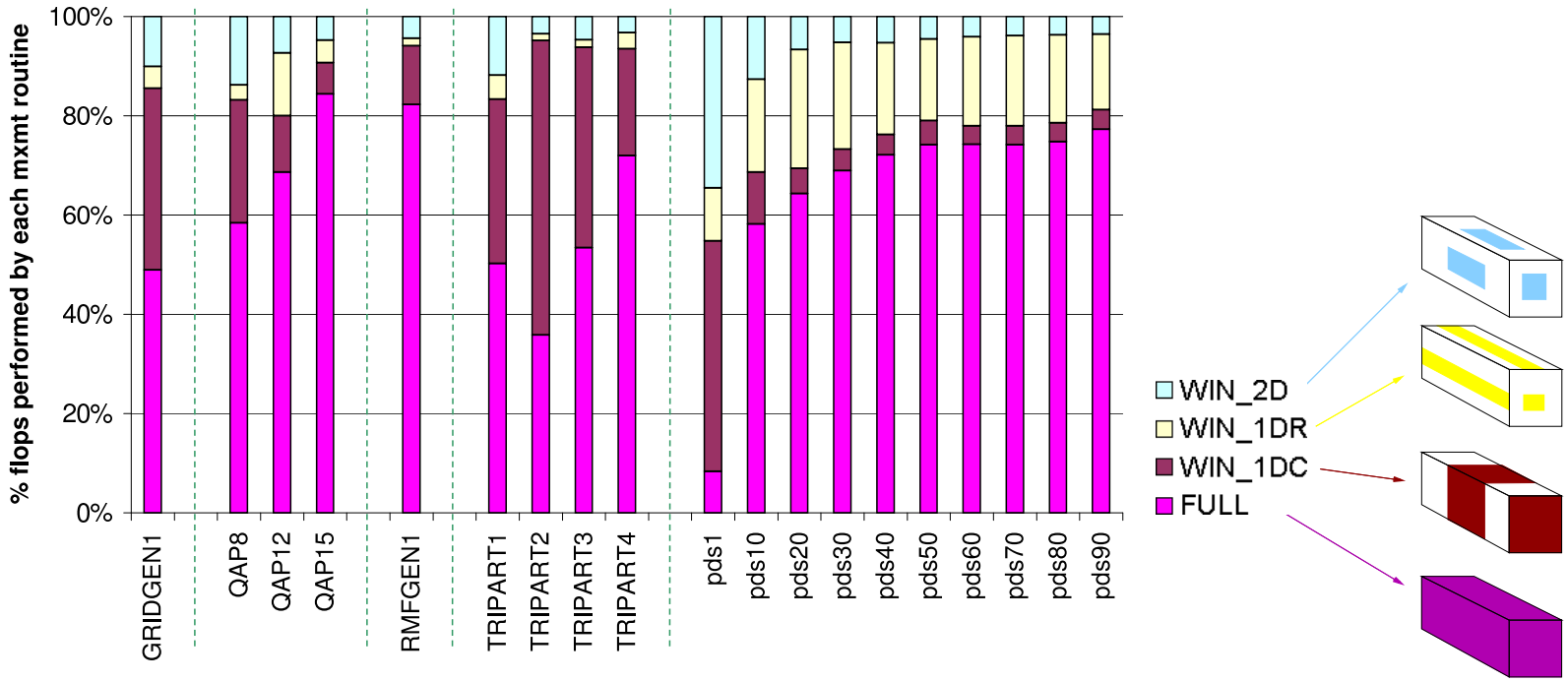
Current work

- Amalgamation
- Blocked sparse Cholesky code within SPLASH-2
  - Get it to work for large matrices
  - Get it to work in parallel

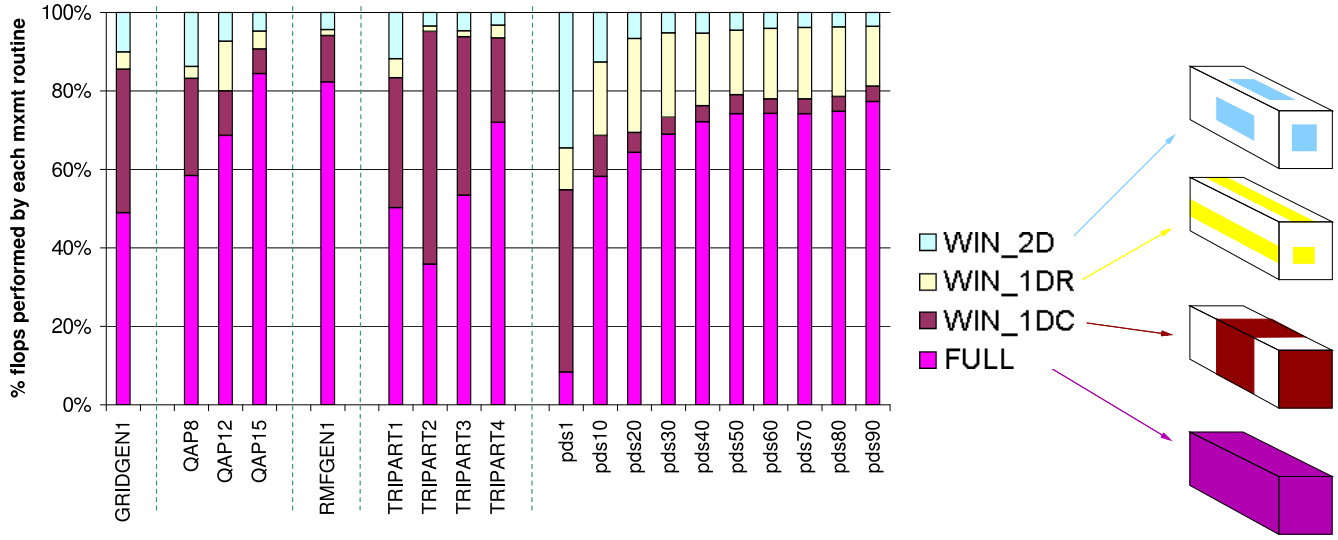
Increase in number of operations in sparse HM Cholesky (4x32 + windows).

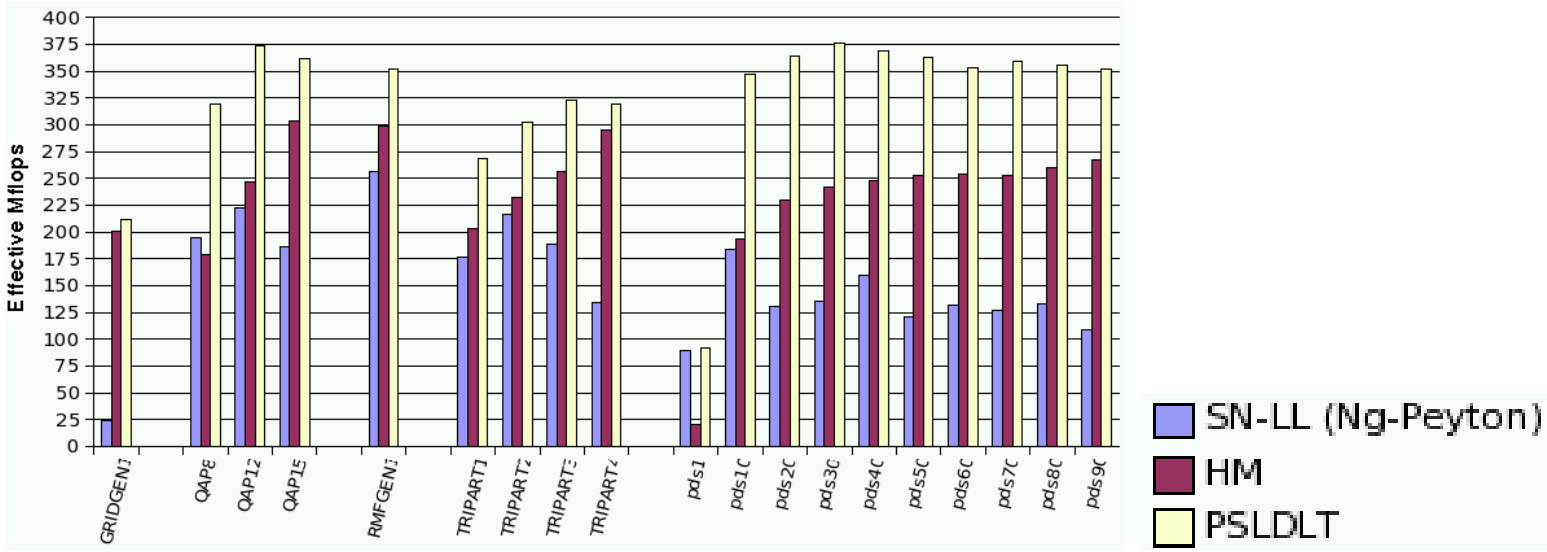


HM flops per MxMt subroutine type



HM flops per MxMt subroutine type





Conclusions & Future Work

- Grouping of rows within submatrices could improve execution
  - Supernodal-HM
- Sparse HM Cholesky is competitive for large problems
  - OOC
- Good chances for exploiting parallelism
  - Parallel version