Scaling Behavior of the Tight Binding Molecular Dynamics Code with Parallel Matrix Diagonalization (ScaLAPACK): Application to Carbon Nanotube

Sep 11, 2007 HLRS, Stuttgart

Dr. Cem Özdoğan

Department of Computer Engineering, Çankaya University,

06530 Ankara, Turkey E-mail:ozdogan@cankaya.edu.tr

Introduction 1

- Describing the motion of N electrons in the field of M fixed nuclear point charges is a central problem in quantum chemistry.

Introduction 1

- Describing the motion of N electrons in the field of M fixed nuclear point charges is a central problem in quantum chemistry.
- Scaling of run time and memory requirements with the studied system sizes N

Introduction 1

- Describing the motion of N electrons in the field of M fixed nuclear point charges is a central problem in quantum chemistry.
- Scaling of run time and memory requirements with the studied system sizes N
  - $O(N^2)$ for semi-emprical methods

Introduction 1

- Describing the motion of N electrons in the field of M fixed nuclear point charges is a central problem in quantum chemistry.
- Scaling of run time and memory requirements with the studied system sizes N
  - $O(N^2)$ for semi-emprical methods
  - $O(N^3)$, $O(N^4)$ for *ab initio* methods

Introduction 1

- Describing the motion of N electrons in the field of M fixed nuclear point charges is a central problem in quantum chemistry.
- Scaling of run time and memory requirements with the studied system sizes N
  - $O(N^2)$ for semi-emprical methods
  - $O(N^3)$, $O(N^4)$ for *ab initio* methods
- The electronic structure of the simulated system will be studied by a Tight Binding (TB) Hamiltonian.

Introduction 1

- Describing the motion of N electrons in the field of M fixed nuclear point charges is a central problem in quantum chemistry.
- Scaling of run time and memory requirements with the studied system sizes N
  - $O(N^2)$ for semi-emprical methods
  - $O(N^3)$, $O(N^4)$ for *ab initio* methods
- The electronic structure of the simulated system will be studied by a Tight Binding (TB) Hamiltonian.
- It is observed that almost 80% percentage of the total simulation time is spent for the diagonalization stage of this TB Hamiltanian to obtain eigenvalues and eigenvectors.

Introduction 2

- As a result, we have an $O(N^3)$ scaling with the increasing system size, N.

Introduction 2

- As a result, we have an $O(N^3)$ scaling with the increasing system size, N.
- The possible solution;

Introduction 2

- As a result, we have an $O(N^3)$ scaling with the increasing system size, N.
- The possible solution;
  - linear scaling method for electronic structure calculations (Phys. Rev. B 67 (3), art. no. 035415, (2003))

Introduction 2

- As a result, we have an $O(N^3)$ scaling with the increasing system size, N.
- The possible solution;
  - linear scaling method for electronic structure calculations (Phys. Rev. B 67 (3), art. no. 035415, (2003))
  - apply a parallel eigen solver for the diagonalization process as an alternative

Introduction 2

- As a result, we have an $O(N^3)$ scaling with the increasing system size, N.
- The possible solution;
  - linear scaling method for electronic structure calculations (Phys. Rev. B 67 (3), art. no. 035415, (2003))
  - apply a parallel eigen solver for the diagonalization process as an alternative
- The application of the SCALAPACK library to a TBMD simulation of Carbon Nonotube.

Introduction 3

- Results;

Introduction 3

- Results;
  - block size of the block cyclic distribution,

Introduction 3

- Results;
  - block size of the block cyclic distribution,
  - processors grid shape,

Introduction 3

- Results;
  - block size of the block cyclic distribution,
  - processors grid shape,
  - scaling behavior,

Introduction 3

- Results;
  - block size of the block cyclic distribution,
  - processors grid shape,
  - scaling behavior,
  - parallel performance metrics such speed-up and efficiency depending on the above parameters

Carbon Nanotube

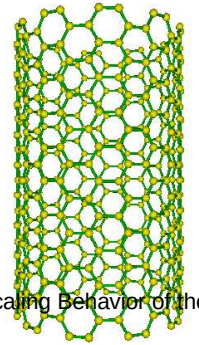
- Carbon nanotubes are thought to play a major role in the design of next generation nanoelectronic and nanoelectromechanical devices due to their remarkable mechanical and electronic properties.

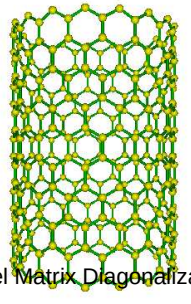
Carbon Nanotube

- Carbon nanotubes are thought to play a major role in the design of next generation nanoelectronic and nanoelectromechanical devices due to their remarkable mechanical and electronic properties.
- SWCNTs are have high flexibility, strength and stiffness, very similar to the properties of individual graphene sheets.

Carbon Nanotube

- Carbon nanotubes are thought to play a major role in the design of next generation nanoelectronic and nanoelectromechanical devices due to their remarkable mechanical and electronic properties.
- SWCNTs are have high flexibility, strength and stiffness, very similar to the properties of individual graphene sheets.
- 10x10 (left) 17x0 (right) (Phys. Rev. B 67 (3), art. no. 035416, (2003))





Linear Scaling

- All the linear scaling methods in the electronic structure calculations methods introduce some approximations and simplifications together with the algorithm complexities.

Linear Scaling

- All the linear scaling methods in the electronic structure calculations methods introduce some approximations and simplifications together with the algorithm complexities.
- This situation also arises in our developed $O(N)$ algorithm as the importance of the buffer size.

Linear Scaling

- All the linear scaling methods in the electronic structure calculations methods introduce some approximations and simplifications together with the algorithm complexities.
- This situation also arises in our developed $O(N)$ algorithm as the importance of the buffer size.
- As a conclusion, these linear scaling algoritms are generally more sensitive to round off errors and accuracy together with the special care of introduced extra parameters/variables.

Linear Scaling

- All the linear scaling methods in the electronic structure calculations methods introduce some approximations and simplifications together with the algorithm complexities.
- This situation also arises in our developed $O(N)$ algorithm as the importance of the buffer size.
- As a conclusion, these linear scaling algoritms are generally more sensitive to round off errors and accuracy together with the special care of introduced extra parameters/variables.
- There are already well-developed packages that making use of linear-scaling methods for electronic structure calculations (SIESTA, CONQUEST, CRYSTAL, ...)

Parallel Eigensolvers

- First thing is to decide whether to use a direct solver, leading to a transformation of the original matrix and thus (for large problems) generating a need for huge main memory, or to use an iterative solver which works with the original matrix. A direct solver consists of the classical three steps:

Parallel Eigensolvers

- First thing is to decide whether to use a direct solver, leading to a transformation of the original matrix and thus (for large problems) generating a need for huge main memory, or to use an iterative solver which works with the original matrix. A direct solver consists of the classical three steps:
  - reduction to symmetric tridiagonal form,

Parallel Eigensolvers

- First thing is to decide whether to use a direct solver, leading to a transformation of the original matrix and thus (for large problems) generating a need for huge main memory, or to use an iterative solver which works with the original matrix. A direct solver consists of the classical three steps:
  - reduction to symmetric tridiagonal form,
  - eigen-decomposition of the tridiagonal matrix,

Parallel Eigensolvers

- First thing is to decide whether to use a direct solver, leading to a transformation of the original matrix and thus (for large problems) generating a need for huge main memory, or to use an iterative solver which works with the original matrix. A direct solver consists of the classical three steps:
  - reduction to symmetric tridiagonal form,
  - eigen-decomposition of the tridiagonal matrix,
  - back-transformation of the eigenvectors.

Parallel Eigensolvers

- First thing is to decide whether to use a direct solver, leading to a transformation of the original matrix and thus (for large problems) generating a need for huge main memory, or to use an iterative solver which works with the original matrix. A direct solver consists of the classical three steps:
  - reduction to symmetric tridiagonal form,
  - eigen-decomposition of the tridiagonal matrix,
  - back-transformation of the eigenvectors.
- The fastest serial algorithm is that based upon Householder reduction to tridiagonal form followed by diagonalization.

Packages 1

- Further many algorithms can easily generate a good guess at the eigenvectors. Unfortunately this is difficult within the Householder methodology.

Packages 1

- Further many algorithms can easily generate a good guess at the eigenvectors. Unfortunately this is difficult within the Householder methodology.
- Basic Linear Algebra Subprograms (BLAS)

Packages 1

- Further many algorithms can easily generate a good guess at the eigenvectors. Unfortunately this is difficult within the Householder methodology.
- Basic Linear Algebra Subprograms (BLAS)
- Linear Algebra Package (LAPACK)

Packages 1

- Further many algorithms can easily generate a good guess at the eigenvectors. Unfortunately this is difficult within the Householder methodology.
- Basic Linear Algebra Subprograms (BLAS)
- Linear Algebra Package (LAPACK)
- PBLAS (Parallel Basic Linear Algebra Subroutines)

Packages 1

- Further many algorithms can easily generate a good guess at the eigenvectors. Unfortunately this is difficult within the Householder methodology.
- Basic Linear Algebra Subprograms (BLAS)
- Linear Algebra Package (LAPACK)
- PBLAS (Parallel Basic Linear Algebra Subroutines)
- Scalable LAPACK (ScaLAPACK)

Packages 1

- Further many algorithms can easily generate a good guess at the eigenvectors. Unfortunately this is difficult within the Householder methodology.
- Basic Linear Algebra Subprograms (BLAS)
- Linear Algebra Package (LAPACK)
- PBLAS (Parallel Basic Linear Algebra Subroutines)
- Scalable LAPACK (ScaLAPACK)
- A (parallel) package for the solution of large eigenvalue problems (P)ARPACK

Packages 2

- Portable, Extensible Toolkit for Scientific computation (PETSc).

Packages 2

- Portable, Extensible Toolkit for Scientific computation (PETSc).
- Parallel Eigensolver System software (PeIGS) library routines.

Packages 2

- Portable, Extensible Toolkit for Scientific computation (PETSc).
- Parallel Eigensolver System software (PeIGS) library routines.
- Parallel LAPACK (PLAPACK).

Packages 2

- Portable, Extensible Toolkit for Scientific computation (PETSc).
- Parallel Eigensolver System software (PeIGS) library routines.
- Parallel LAPACK (PLAPACK).
- Parallel Research on Invariant Subspace Methods - Parallel Symmetric Eigensolver algorithm (PRISM).

Packages 2

- Portable, Extensible Toolkit for Scientific computation (PETSc).
- Parallel Eigensolver System software (PeIGS) library routines.
- Parallel LAPACK (PLAPACK).
- Parallel Research on Invariant Subspace Methods - Parallel Symmetric Eigensolver algorithm (PRISM).
- Parallel Engineering and Scientific Subroutine Library (PESSL).

Packages 2

- Portable, Extensible Toolkit for Scientific computation (PETSc).
- Parallel Eigensolver System software (PeIGS) library routines.
- Parallel LAPACK (PLAPACK).
- Parallel Research on Invariant Subspace Methods - Parallel Symmetric Eigensolver algorithm (PRISM).
- Parallel Engineering and Scientific Subroutine Library (PESSL).
- Within the ScaLAPACK project many LAPACK routines were ported to distributed memory computers using MPI. The basic routines of ScaLAPACK are the PBLAS.

Routines & Algorithms

- They contain parallel versions of the BLAS which are parallelized using BLACS (Basic Linear Algebra Communication Subprograms) for communication and sequential BLAS for computation.

Routines & Algorithms

- They contain parallel versions of the BLAS which are parallelized using BLACS (Basic Linear Algebra Communication Subprograms) for communication and sequential BLAS for computation.
- PDSYEVD. The parallel eigensolver, PDSYEVD, is found in ScaLAPACK and is based upon the divide-and-conquer algorithm.

Routines & Algorithms

- They contain parallel versions of the BLAS which are parallelized using BLACS (Basic Linear Algebra Communication Subprograms) for communication and sequential BLAS for computation.
- PDSYEVD. The parallel eigensolver, PDSYEVD, is found in ScaLAPACK and is based upon the divide-and-conquer algorithm.
- PDSYEVX. The parallel eigensolver, PDSYEVX, is found in PESSL, which is a library of parallel solvers for scientific problems supplied by IBM. We used PDSYEV (Scpk 1.5), PDSYEVX (Scpk 1.5), PDSYEVD (Scpk 1.7).

Routines & Algorithms

- They contain parallel versions of the BLAS which are parallelized using BLACS (Basic Linear Algebra Communication Subprograms) for communication and sequential BLAS for computation.
- PDSYEVD. The parallel eigensolver, PDSYEVD, is found in ScaLAPACK and is based upon the divide-and-conquer algorithm.
- PDSYEVX. The parallel eigensolver, PDSYEVX, is found in PESSL, which is a library of parallel solvers for scientific problems supplied by IBM. We used PDSYEV (Scpk 1.5), PDSYEVX (Scpk 1.5), PDSYEVD (Scpk 1.7).
- PMR3, PBDn, one-sided Block Factored Jacobi implementation,PRISM.

Hamiltonian Matrix

$$H = \begin{bmatrix} ints.\ of\ 1^{st}\ e^{-}\ of\ 1^{st}\ atom\ with\ all\ neighbors \\ ints.\ of\ 2^{nd}\ e^{-}\ of\ 1^{st}\ atom\ with\ all\ neighbors \\ \vdots \\ ints.\ of\ 4*N^{th}\ e^{-}\ of\ 4*n^{th}\ atom\ with\ all\ neighbors \end{bmatrix}$$

- This matrix is a upper triangle matrix, having nonzero terms only in upper triangle. This is necessary for diagonalization procedure.

Hamiltonian Matrix

$$H = \begin{bmatrix} ints.\ of\ 1^{st}\ e^{-}\ of\ 1^{st}\ atom\ with\ all\ neighbors \\ ints.\ of\ 2^{nd}\ e^{-}\ of\ 1^{st}\ atom\ with\ all\ neighbors \\ \vdots \\ ints.\ of\ 4 * N^{th}\ e^{-}\ of\ 4 * n^{th}\ atom\ with\ all\ neighbors \end{bmatrix}$$

- This matrix is a upper triangle matrix, having nonzero terms only in upper triangle. This is necessary for diagonalization procedure.
- After diagonalization, band structure energy is computed.

Block Cyclic Distribution

	0		1		0		1		0
0	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}
	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}	a_{28}	a_{29}
	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}	a_{37}	a_{38}	a_{39}
1	a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}	a_{49}
	a_{51}	a_{52}	a_{53}	a_{54}	a_{55}	a_{56}	a_{57}	a_{58}	a_{59}
	a_{61}	a_{62}	a_{63}	a_{64}	a_{65}	a_{66}	a_{67}	a_{68}	a_{69}
0	a_{71}	a_{72}	a_{73}	a_{74}	a_{75}	a_{76}	a_{77}	a_{78}	a_{79}
	a_{81}	a_{82}	a_{83}	a_{84}	a_{85}	a_{86}	a_{87}	a_{88}	a_{89}
	a_{91}	a_{92}	a_{93}	a_{94}	a_{95}	a_{96}	a_{97}	a_{98}	a_{99}

Figure 1: Block cyclic 2D distribution of a 9x9 matrix subdivided into 3x2 blocks on a 2x2 processor grid.

- The numbers outside the matrix indicate processor row and column indices, respectively.

Code Segment

```
CALL BLACS_PINFO( iam, nproc )
!     Initialize a single BLACS context
CALL BLACS_GET( -1, 0, CONTEXT )
CALL BLACS_GRIDINIT( CONTEXT, 'R', NPROW, NPCOL )
CALL BLACS_GRIDINFO( CONTEXT, NPROW, NPCOL, MYROW, MYCOL )
IF( MYROW.EQ.-1 )  GO TO 20
mxlocr= NUMROC(r,rb,MYROW,rsrc,NPROW)
mxlocc= NUMROC(c,cb,MYCOL,csrc,NPCOL)
! Local leading dimension of A--> hhwork
llda = max(mxlocr,mxlocc)
allocate(hhwork(1:llda,1:llda),z(1:llda,1:llda),stat=allocatestatus)
CALL DESCINIT( DESCA, r, c, rb, cb, 0, 0, CONTEXT,llda, INFO )
CALL DESCINIT( DESCZ, r, c, rb, cb, 0, 0, CONTEXT,llda, INFO )
!     Ask PDSYEV to compute the entire eigendecomposition
jobz='V'
uplo='U'
CALL PDSYEV( jobz,uplo,nhdim, z, 1, 1, DESCA, eval, z, 1,1,DESCZ, WORK, -1, INFO )
lwork=work(1)
allocate(work(1:lwork),stat=allocatestatus)
CALL PDSYEV( jobz,uplo,nhdim, hhwork, 1, 1, DESCA, eval, z, 1, 1,DESCZ, WORK,
            LWORK, INFO )
CALL PDLAPRNTc( r, c, z, 1, 1, DESCZ, 0, 0, 'hh', hh, WORK )
deallocate(work,stat=allocatestatus)
deallocate(z,stat=allocatestatus)
deallocate(hhwork,stat=allocatestatus)
CALL BLACS_GRIDEXIT( CONTEXT )
```

Sparsity

Table 1: Sparsity of the Hamiltonian Matrix. The criteria is taken as $\leq 10^{-6}$.
????

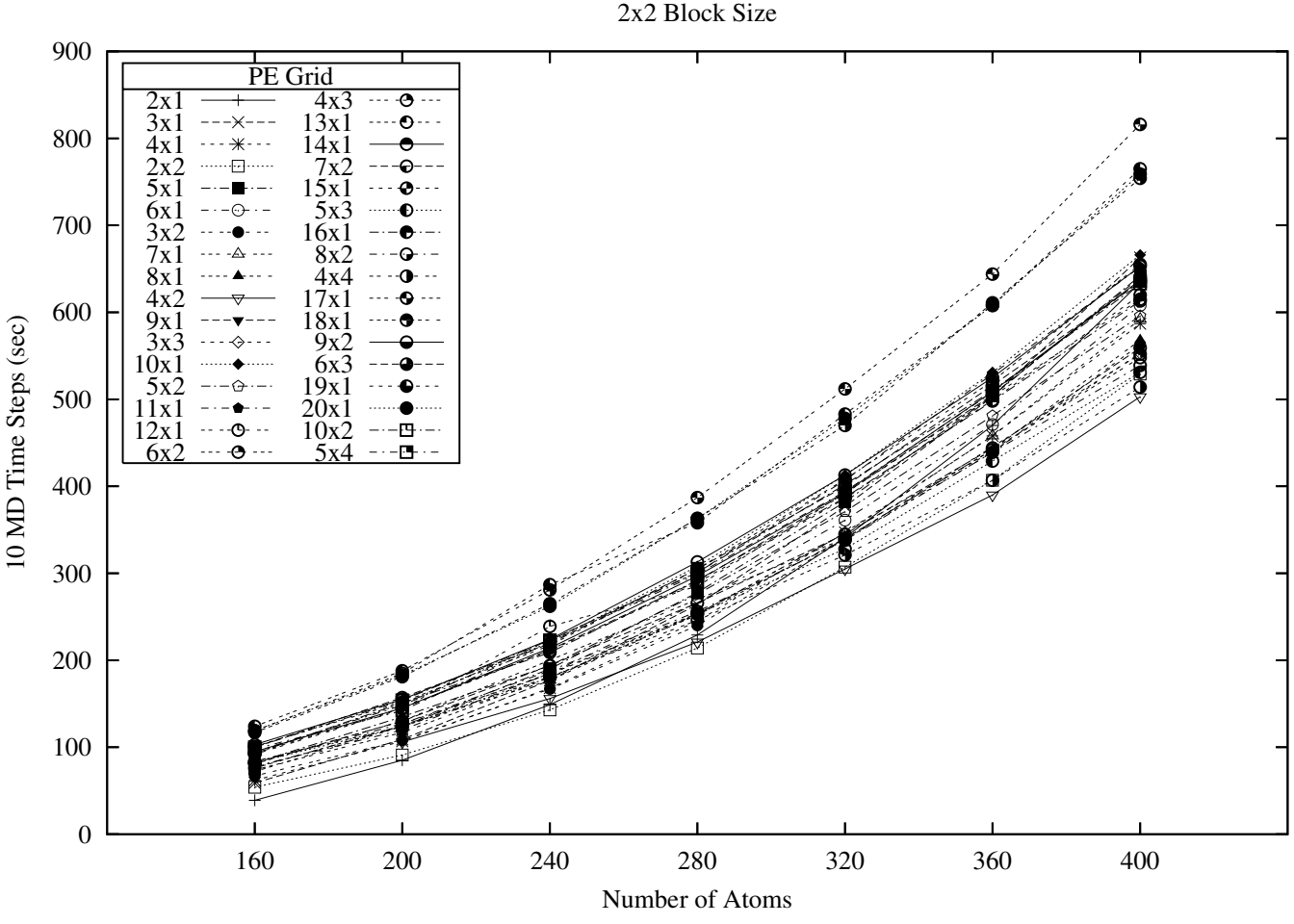
nlay	ntom	nhdim	jirhh	lda	jirlda	lda	jirlda	lda	jirlda	$\frac{jirhh}{nhdim^2} * 100$	$\frac{jirlda}{lda^2} * 100$
28	560	2241	2209	576	576	576	576	640	640	0.04	0.16
32	640	2561	2529	641	641	641	641	641	641	0.04	0.16
36	720	2881	2849	736	736	768	768	768	768	0.03	0.13
40	800	3201	3169	801	801	832	832	896	896	0.03	0.11
44	880	3521	3489	896	896	896	896	896	896	0.03	0.11
48	960	3841	3809	961	961	961	961	1024	1024	0.03	0.1
52	1040	4161	4129	1056	1056	1088	1088	1089	1089	0.02	0.09
56	1120	4481	4449	-	-	1152	1152	1152	1152	0.02	0.09
60	1200	4801	4769	1216	1216	1216	1216	1280	1280	0.02	0.08
64	1280	5121	5089	1281	1281	1281	1281	1281	1281	0.02	0.08
68	1360	5441	5409	1376	1376	1408	1408	1408	1408	0.02	0.07
72	1440	5761	5729	1441	1441	1472	1472	1536	1536	0.02	0.07
76	1520	6081	6049	1536	1536	1536	1536	1536	1536	0.02	0.07
80	1600	6401	6369	1601	1601	1601	1601	1664	1664	0.02	0.06
84	1680	6721	6689	1696	1696	1728	1728	1729	1729	0.01	0.06
88	1760	7041	7009	1761	1761	-	-	1792	1792	0.01	0.06
92	1840	7361	7329	1856	1856	1856	1856	1920	1920	0.01	0.05
96	1920	7681	7649	1921	1921	1921	1921	1921	1921	0.01	0.05
100	2000	8001	7969	2016	2016	2048	2048	2048	2048	0.01	0.05
104	2080	8321	8289	2081	2081	2112	2112	2176	2176	0.01	0.05
108	2160	8641	8609	2176	2176	2176	2176	2176	2176	0.01	0.05
112	2240	8961	8929	2241	2241	2241	2241	2304	2304	0.01	0.04

Memory Usage

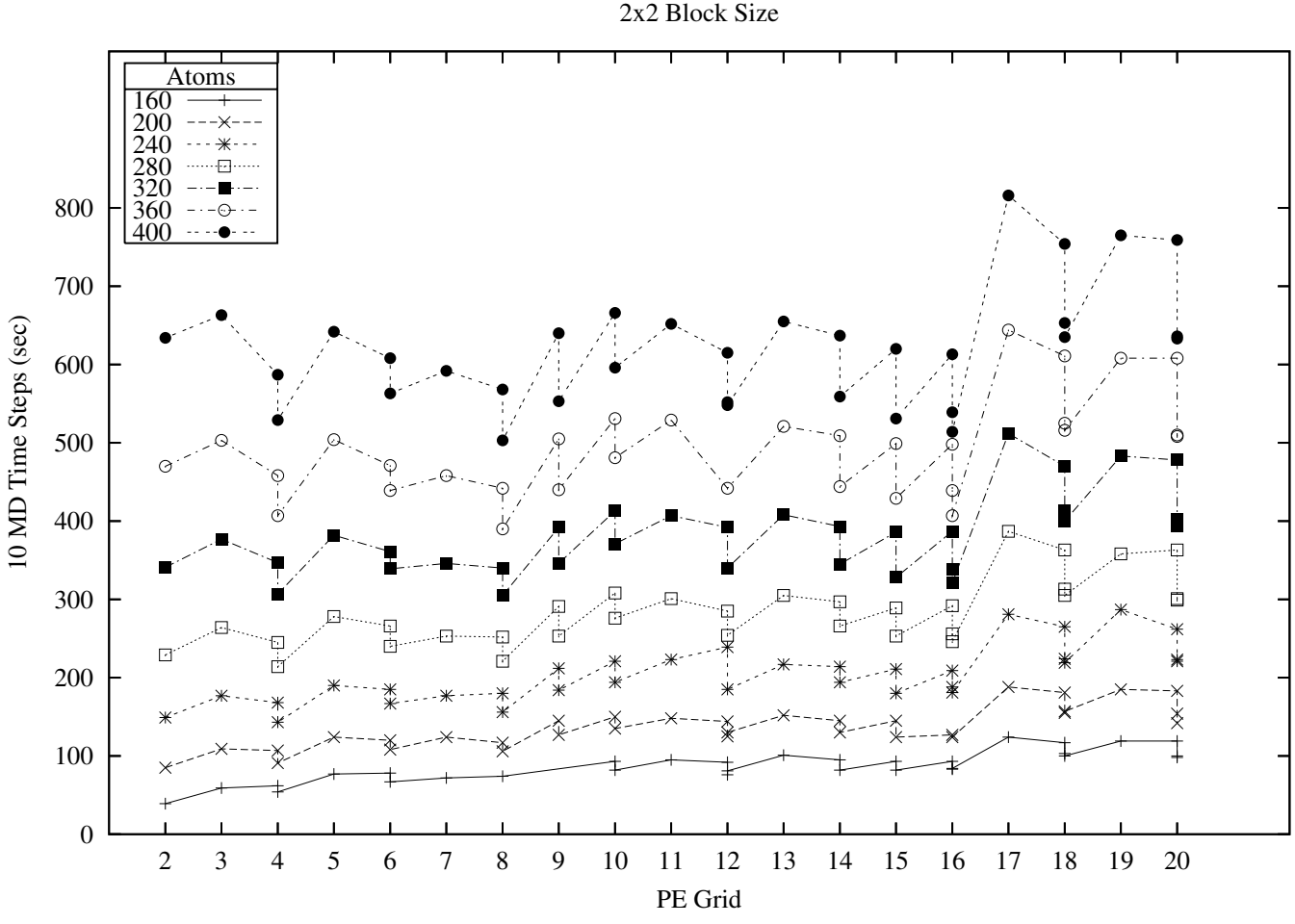
Table 1: Memory Requirements of the Diagonlization Routine (values are in MB)

HH	Work	HHWork+Z	HHWork+Z+Work	All
38.28	3.4	6.25	9.65	47.94
50	3.72	6.27	9.99	60
63.28	4.3	9	13.3	76.59
78.12	4.75	12.25	17	95.14
94.53	5.07	12.25	17.32	111.87
112.5	9.4	16	25.4	137.91
132.03	10.1	18.1	28.19	160.24
153.12	10.8	20.25	31.05	184.19
175.78	11.69	25	36.69	212.48
200	12.32	25.04	37.36	237.38
225.78	13.15	30.25	43.4	269.2
253.12	13.97	36	49.97	303.12
282.03	14.61	36	50.61	332.66
312.5	15.5	42.25	57.75	370.27
344.53	16.2	45.62	61.81	406.37
378.12	16.9	49	65.9	444.05
413.28	24.97	56.25	81.22	494.53
450	25.92	56.31	82.23	532.26
488.28	27	64	91	579.31
528.12	28.2	72.25	100.45	628.6
569.53	29.15	72.25	101.4	670.96
612.5	30.22	81	111.22	723.75

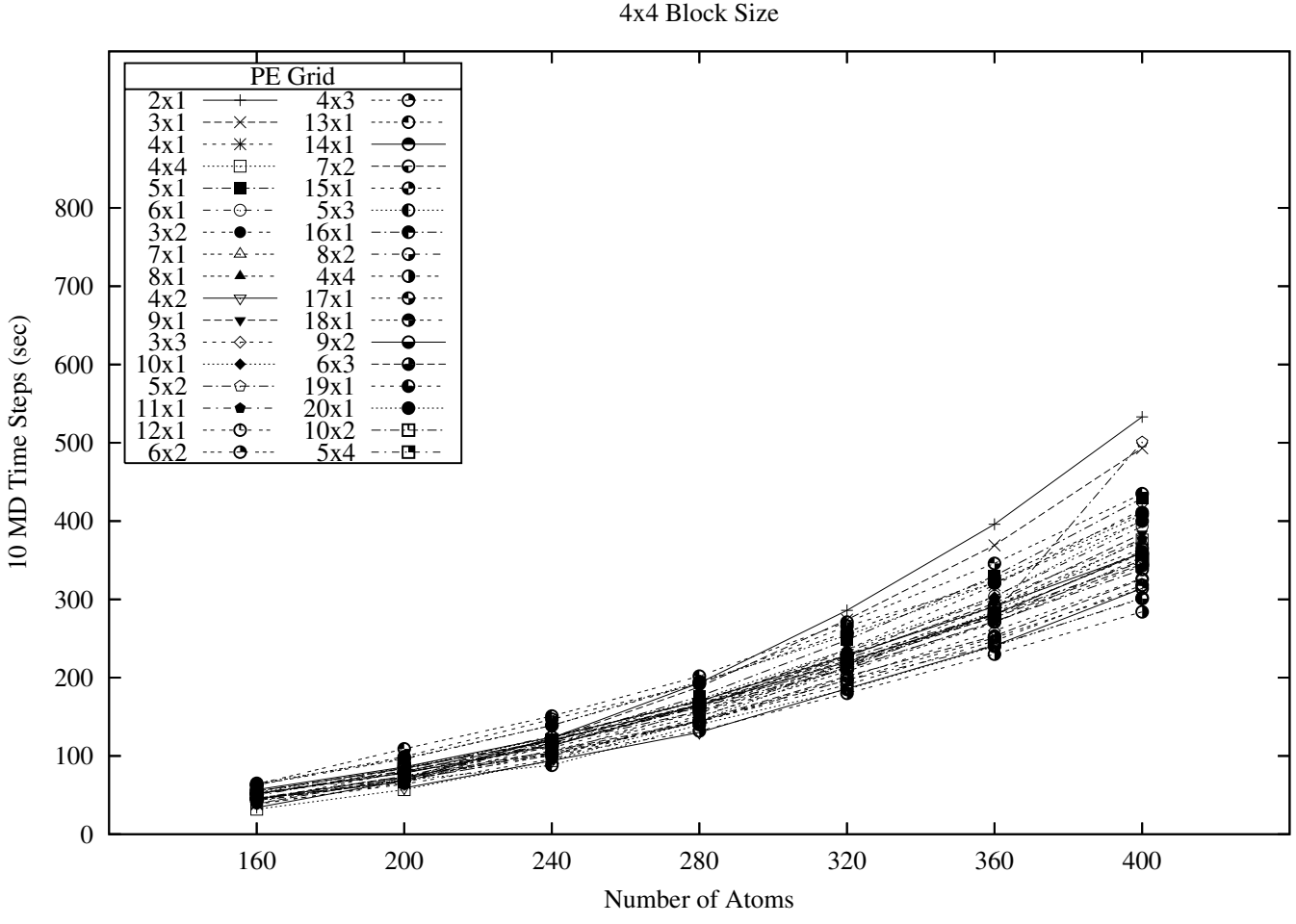
Results 1


2x2 Block Size
PE Grid
2x1
3x1
4x1
2x2
5x1
6x1
3x2
7x1
8x1
4x2
9x1
3x3
10x1
5x2
11x1
12x1
6x2
4x3
13x1
14x1
7x2
15x1
5x3
16x1
8x2
4x4
17x1
18x1
9x2
6x3
19x1
20x1
10x2
5x4
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
900
160
200
240
280
320
360
400
Number of Atoms

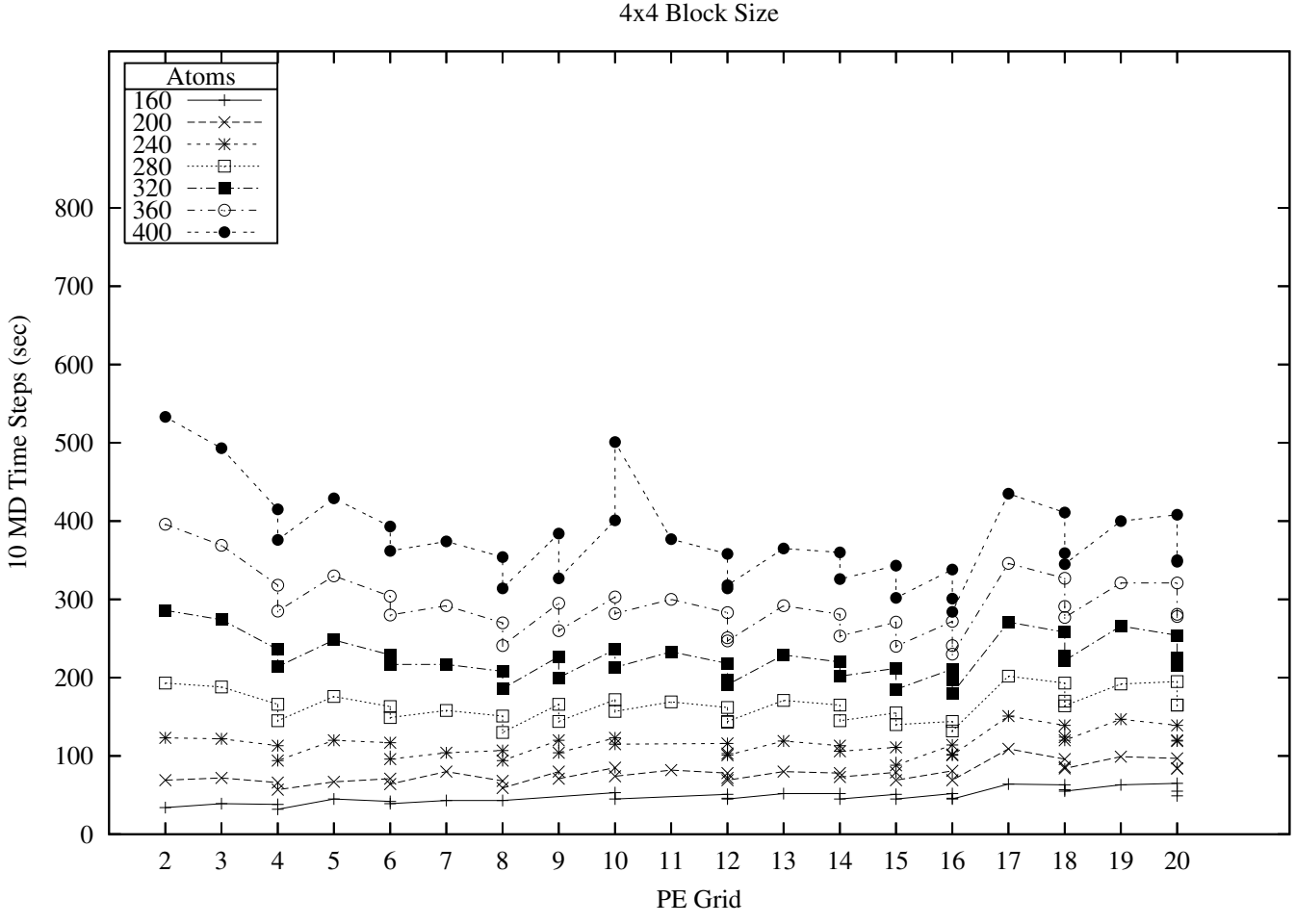
Results 2


2x2 Block Size
Atoms
160
200
240
280
320
360
400
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PE Grid

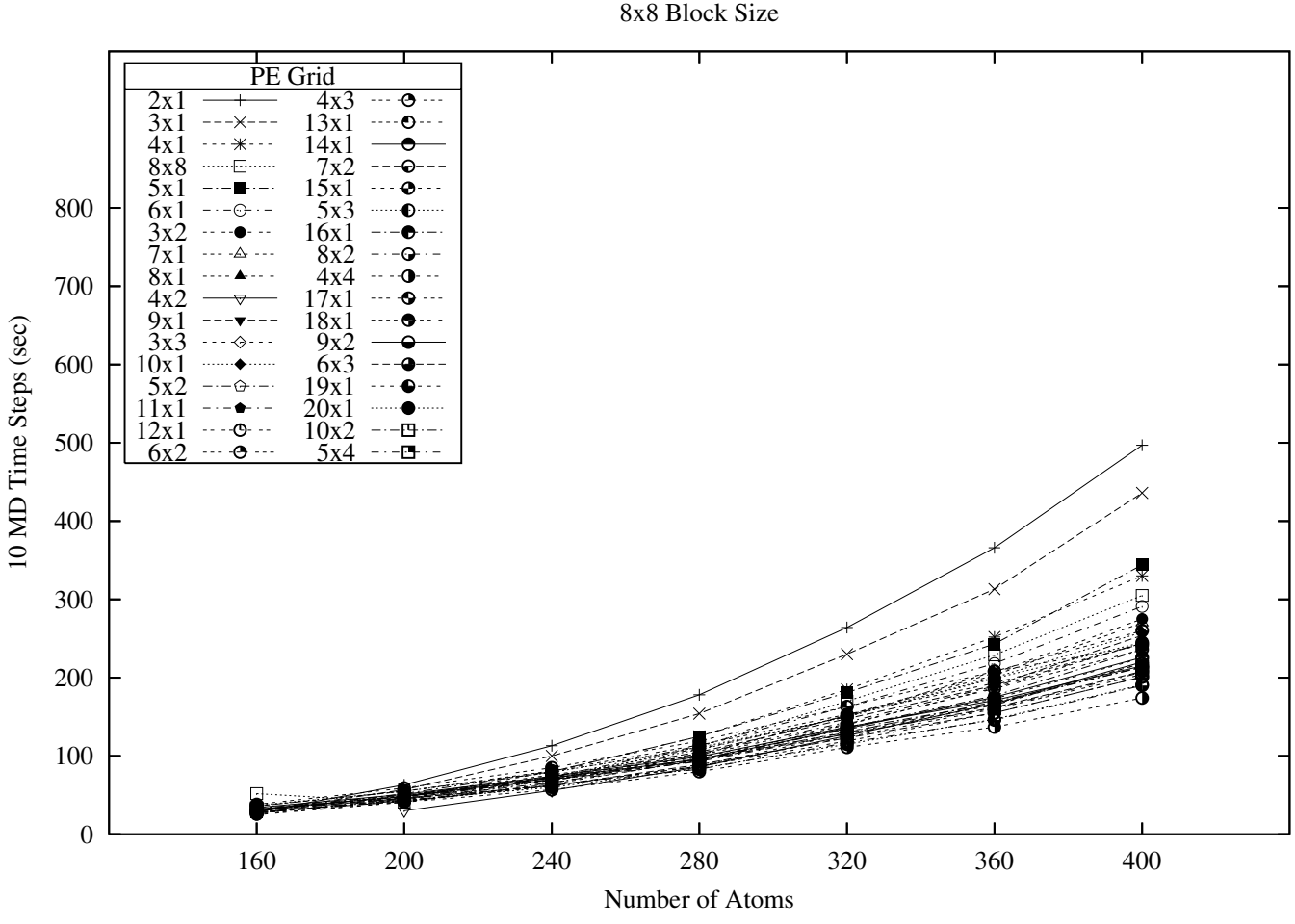
Results 3


4x4 Block Size
PE Grid
2x1
3x1
4x1
4x4
5x1
6x1
3x2
7x1
8x1
4x2
9x1
3x3
10x1
5x2
11x1
12x1
6x2
4x3
13x1
14x1
7x2
15x1
5x3
16x1
8x2
4x4
17x1
18x1
9x2
6x3
19x1
20x1
10x2
5x4
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
160
200
240
280
320
360
400
Number of Atoms

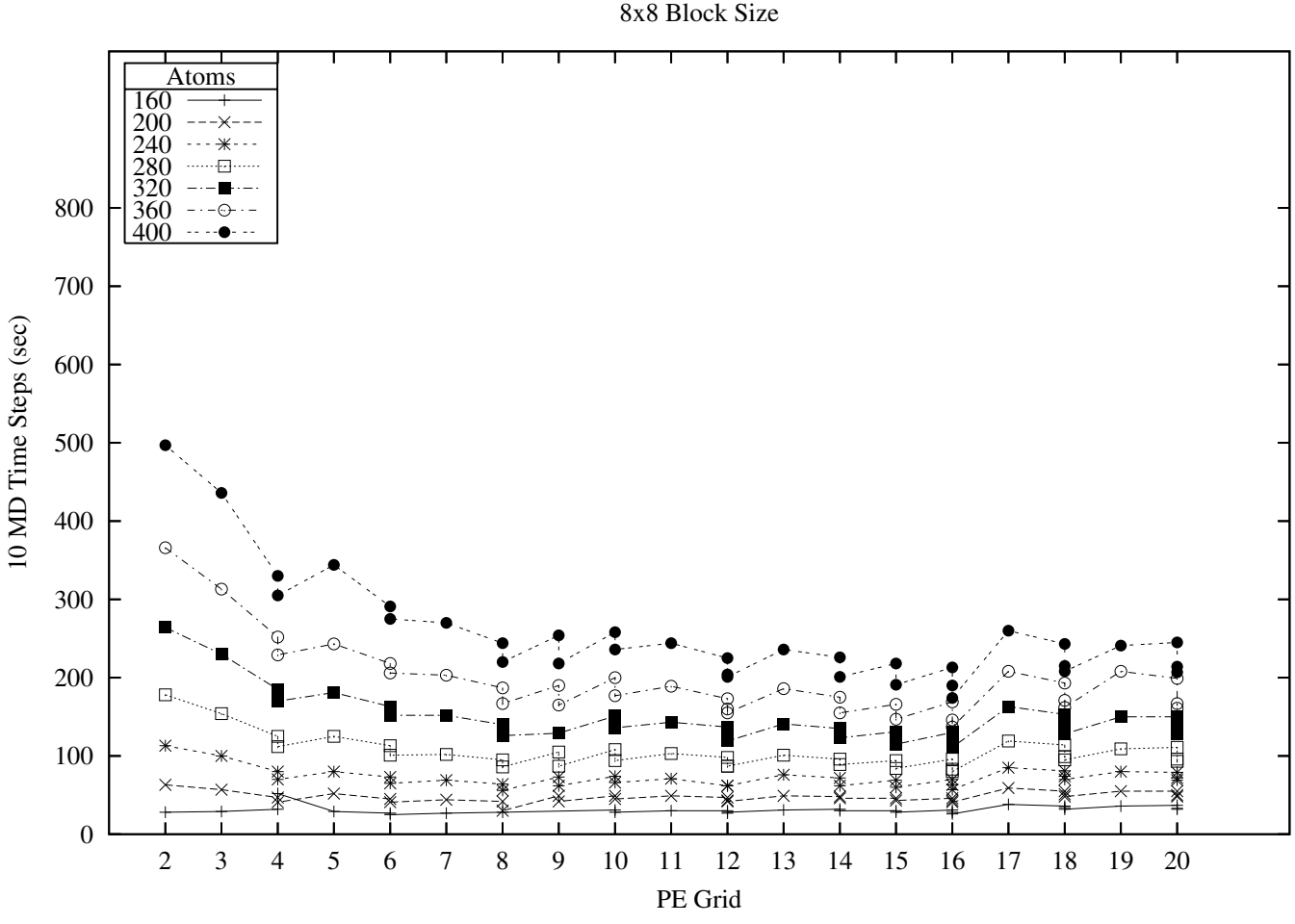
Results 4


4x4 Block Size
Atoms
160
200
240
280
320
360
400
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PE Grid

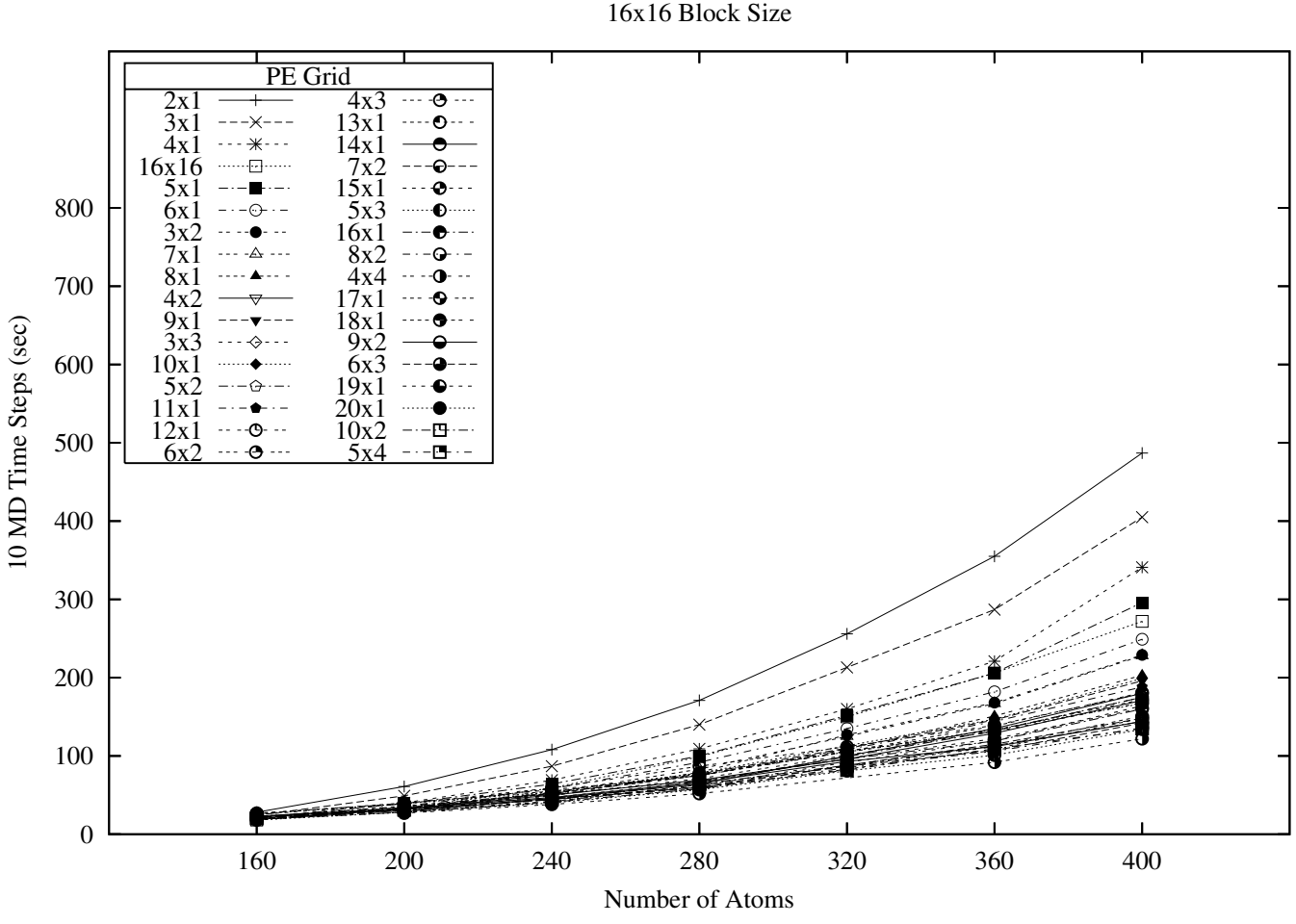
Results 5


8x8 Block Size
PE Grid
2x1
3x1
4x1
8x8
5x1
6x1
3x2
7x1
8x1
4x2
9x1
3x3
10x1
5x2
11x1
12x1
6x2
4x3
13x1
14x1
7x2
15x1
5x3
16x1
8x2
4x4
17x1
18x1
9x2
6x3
19x1
20x1
10x2
5x4
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
160
200
240
280
320
360
400
Number of Atoms

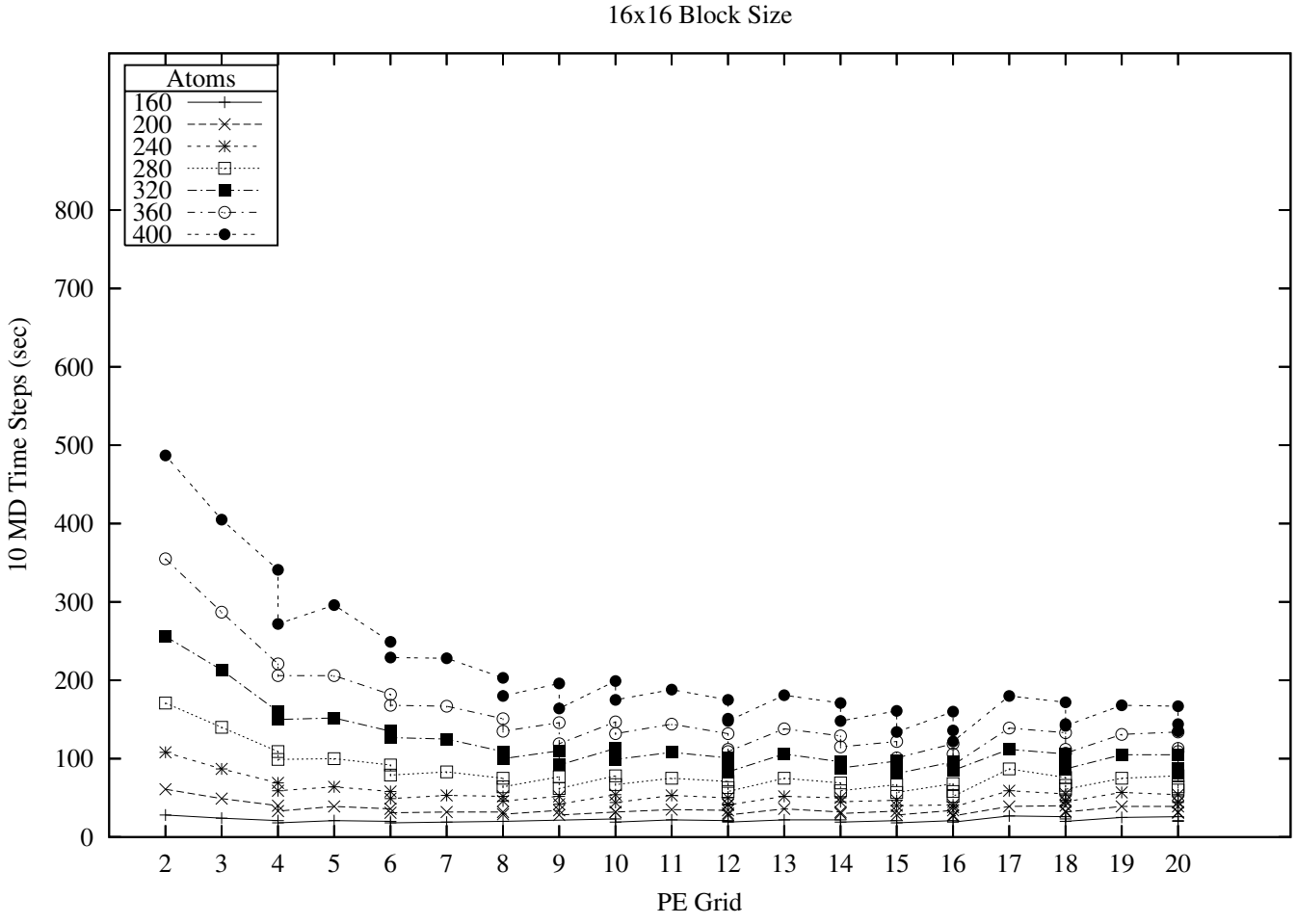
Results 6


8x8 Block Size
Atoms
160
200
240
280
320
360
400
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PE Grid

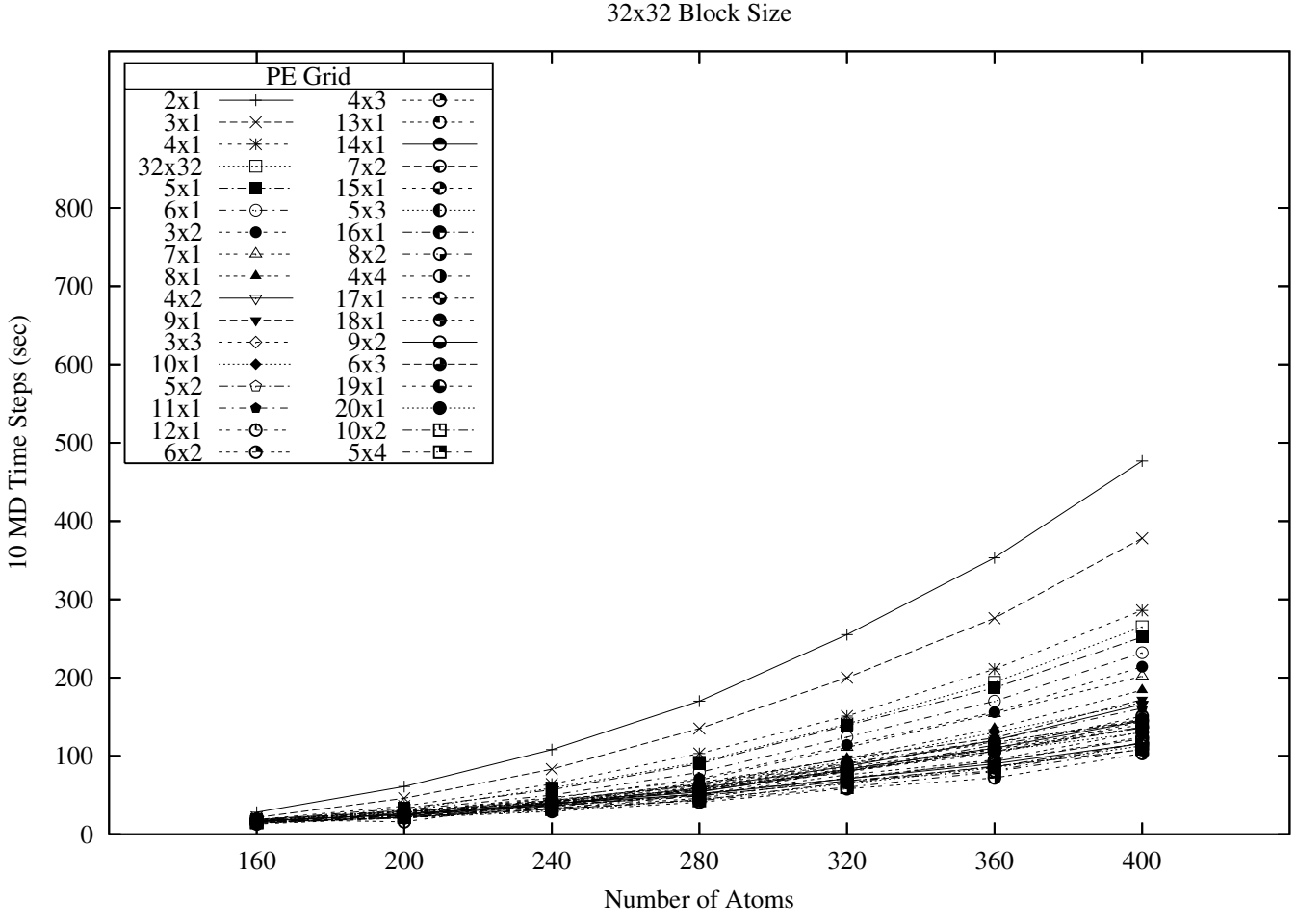
Results 7


16x16 Block Size
PE Grid
2x1
3x1
4x1
16x16
5x1
6x1
3x2
7x1
8x1
4x2
9x1
3x3
10x1
5x2
11x1
12x1
6x2
4x3
13x1
14x1
7x2
15x1
5x3
16x1
8x2
4x4
17x1
18x1
9x2
6x3
19x1
20x1
10x2
5x4
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
160
200
240
280
320
360
400
Number of Atoms

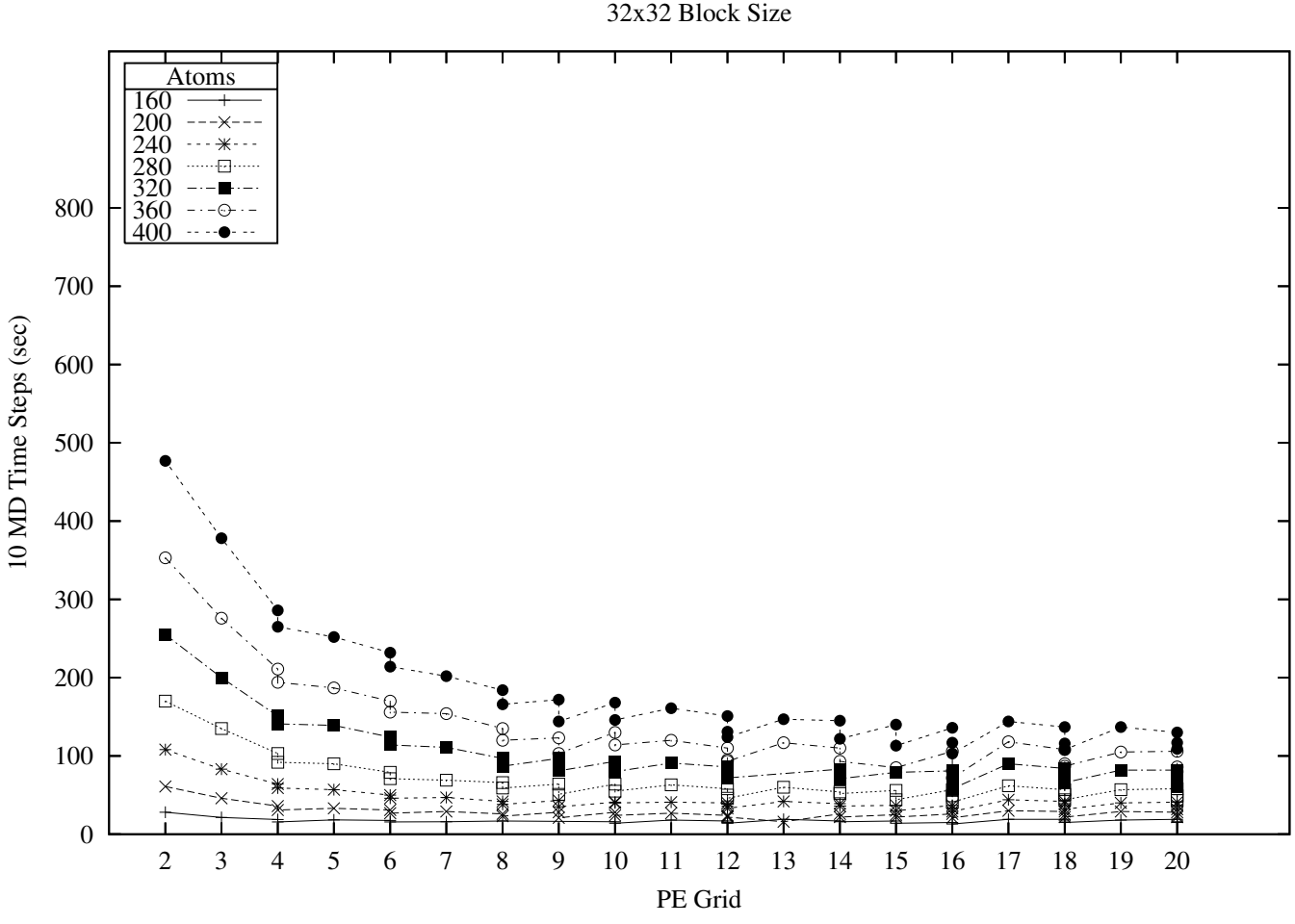
Results 8


16x16 Block Size
Atoms
160
200
240
280
320
360
400
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PE Grid

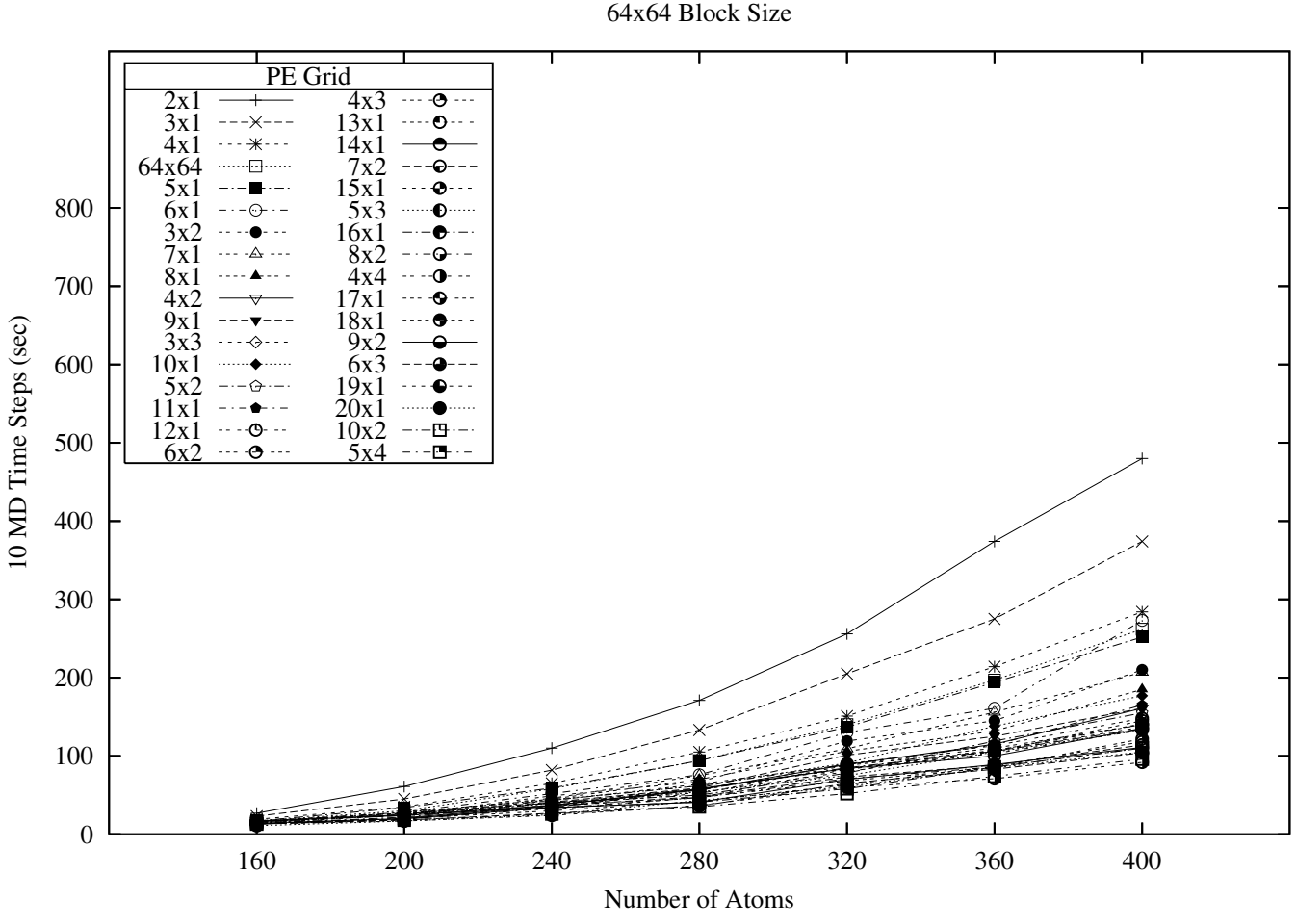
Results 9


32x32 Block Size
PE Grid
2x1
3x1
4x1
32x32
5x1
6x1
3x2
7x1
8x1
4x2
9x1
3x3
10x1
5x2
11x1
12x1
6x2
4x3
13x1
14x1
7x2
15x1
5x3
16x1
8x2
4x4
17x1
18x1
9x2
6x3
19x1
20x1
10x2
5x4
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
160
200
240
280
320
360
400
Number of Atoms

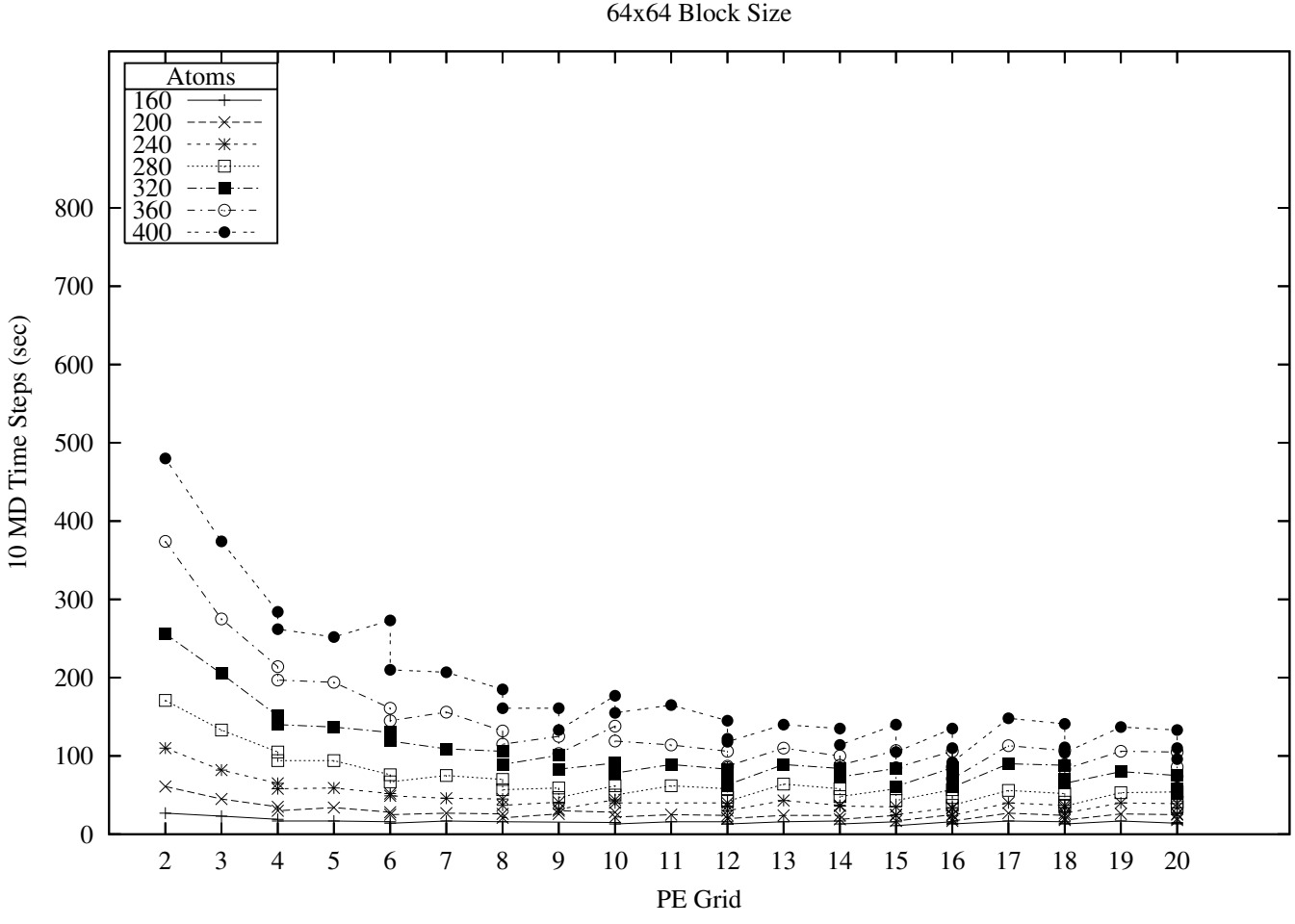
Results 10


32x32 Block Size
Atoms
160
200
240
280
320
360
400
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PE Grid

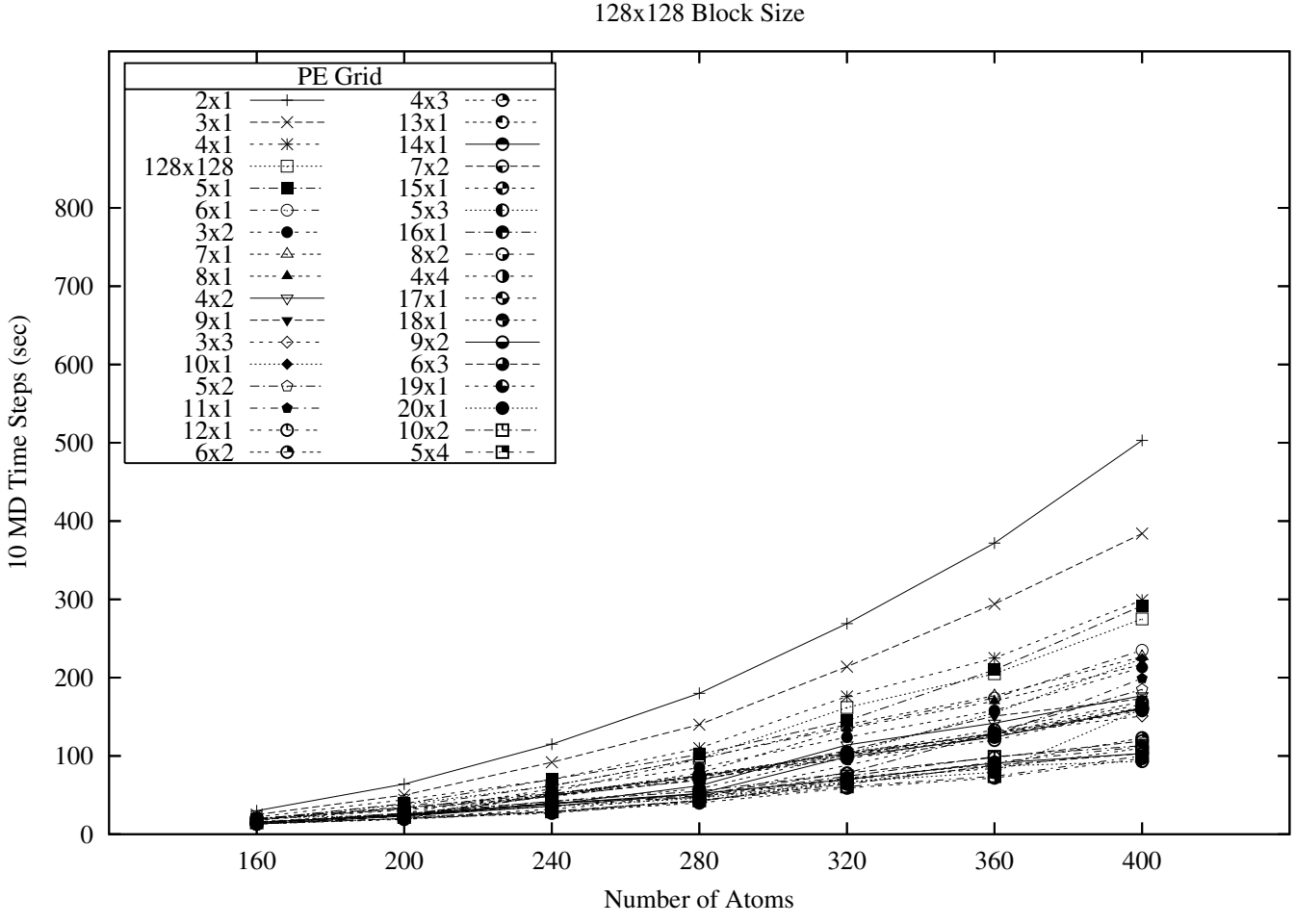
Results 11


64x64 Block Size
PE Grid
2x1
3x1
4x1
64x64
5x1
6x1
3x2
7x1
8x1
4x2
9x1
3x3
10x1
5x2
11x1
12x1
6x2
4x3
13x1
14x1
7x2
15x1
5x3
16x1
8x2
4x4
17x1
18x1
9x2
6x3
19x1
20x1
10x2
5x4
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
160
200
240
280
320
360
400
Number of Atoms

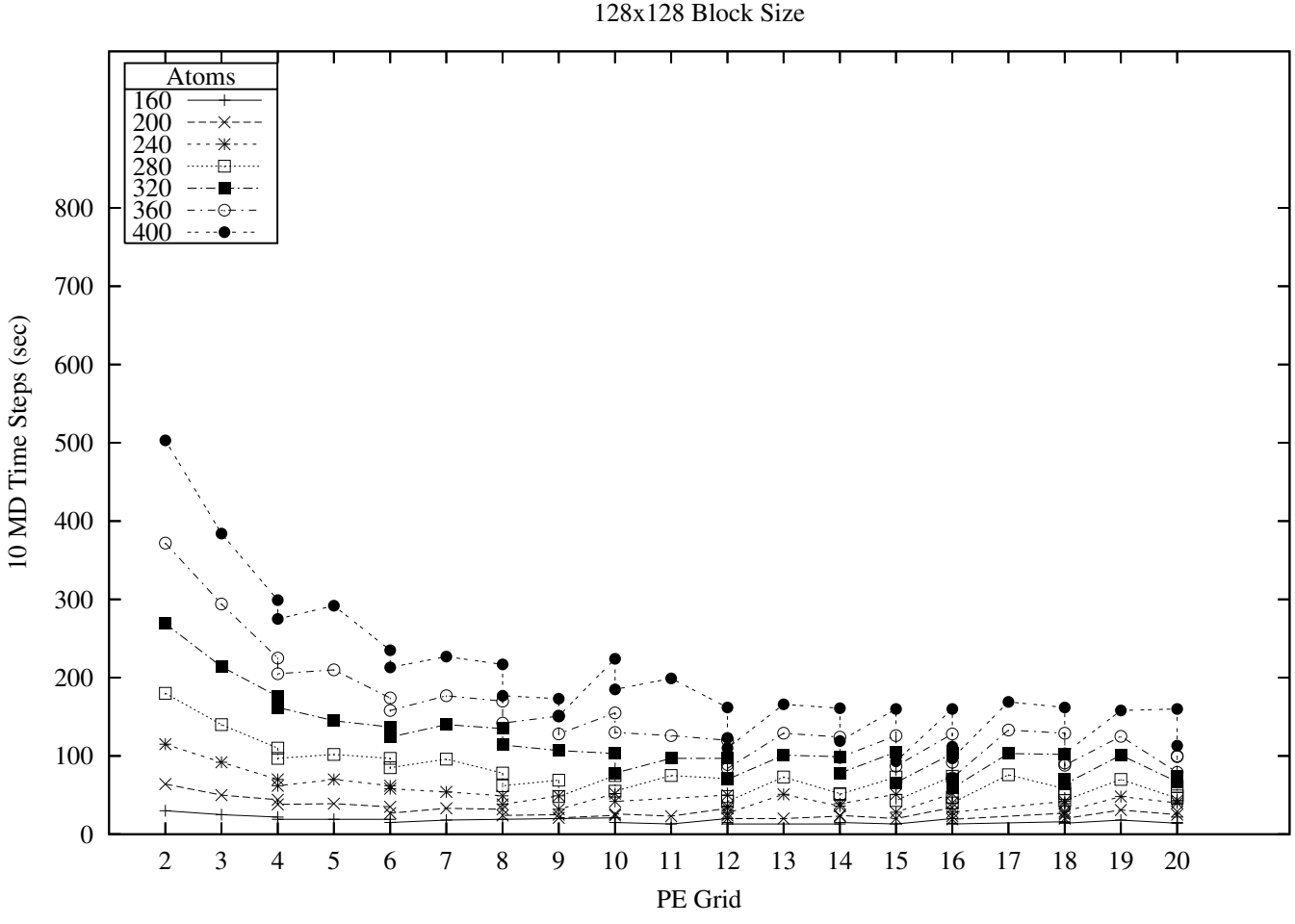
Results 12


64x64 Block Size
Atoms
160
200
240
280
320
360
400
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PE Grid

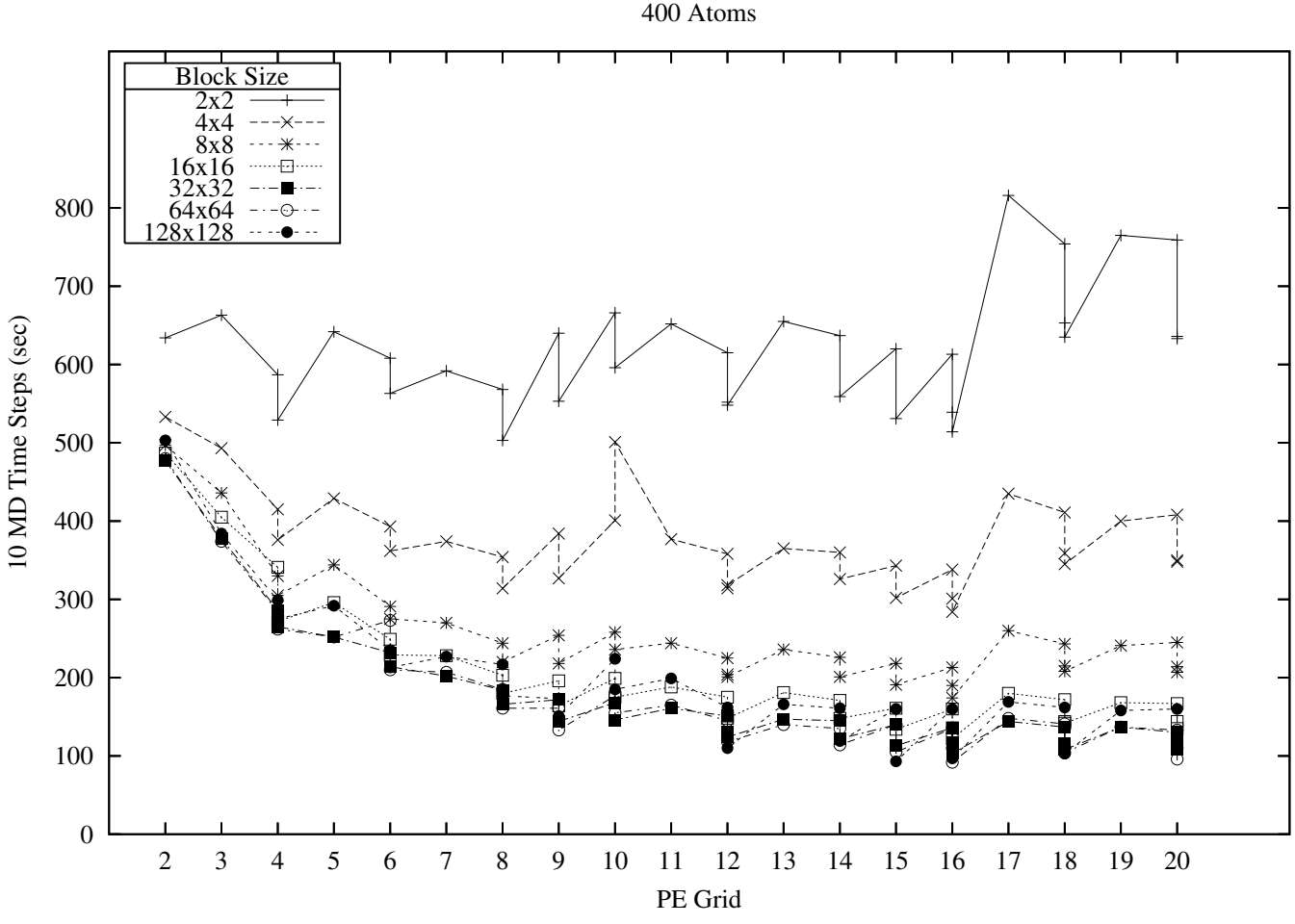
Results 13


128x128 Block Size
PE Grid
2x1
3x1
4x1
128x128
5x1
6x1
3x2
7x1
8x1
4x2
9x1
3x3
10x1
5x2
11x1
12x1
6x2
4x3
13x1
14x1
7x2
15x1
5x3
16x1
8x2
4x4
17x1
18x1
9x2
6x3
19x1
20x1
10x2
5x4
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
160
200
240
280
320
360
400
Number of Atoms

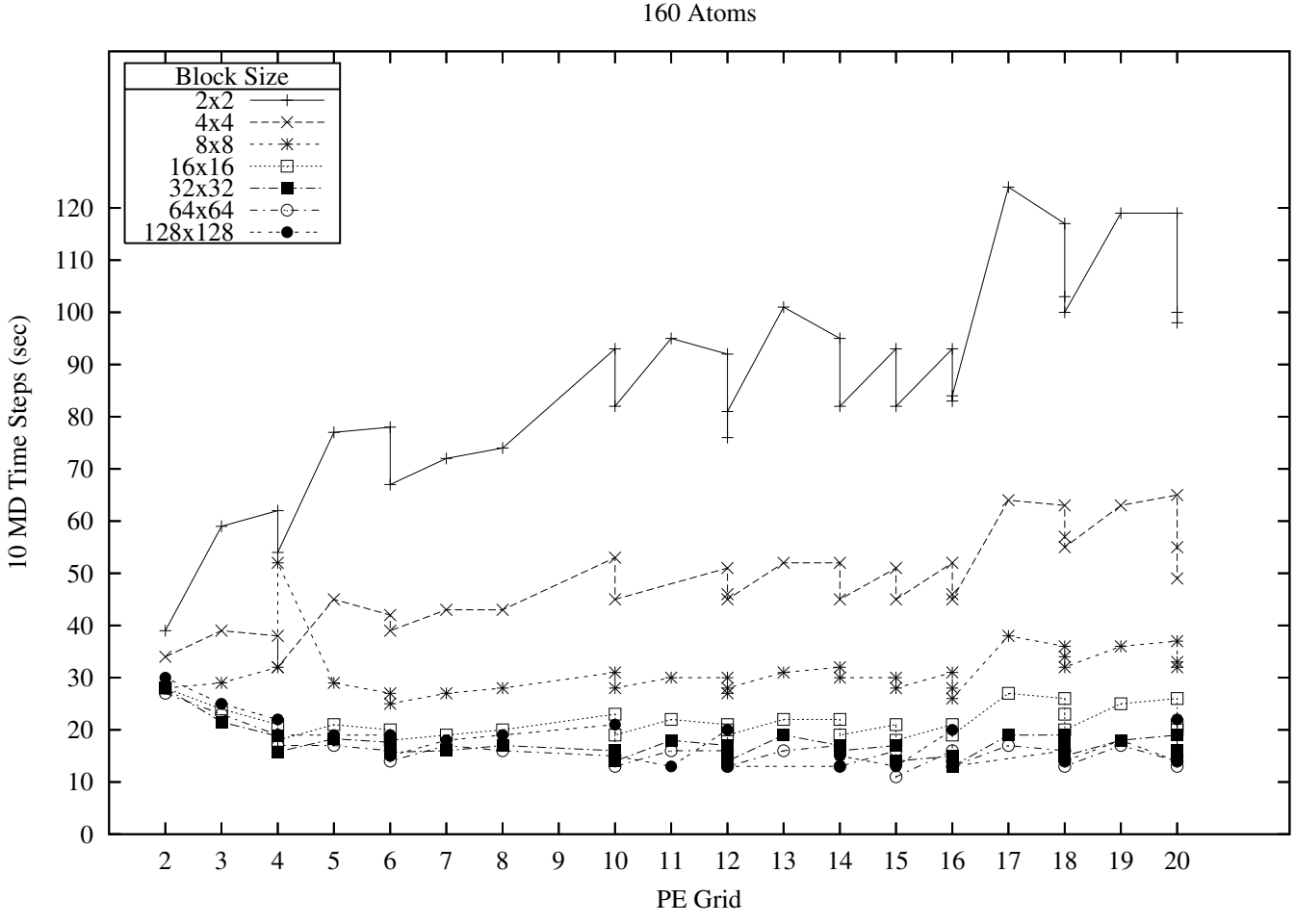
Results 14


128x128 Block Size
Atoms
160
200
240
280
320
360
400
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PE Grid

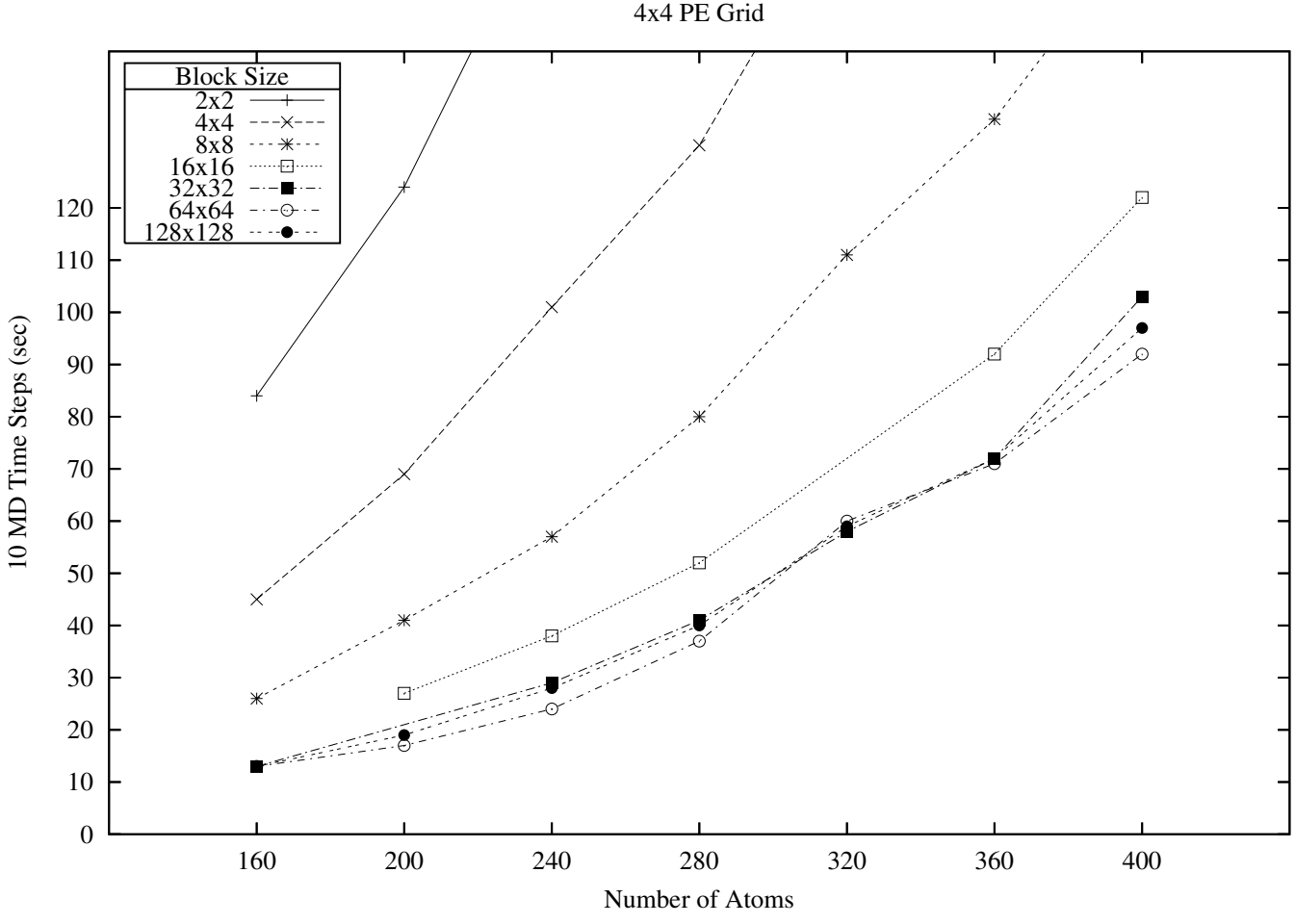
Results 15


400 Atoms
Block Size
2x2
4x4
8x8
16x16
32x32
64x64
128x128
10 MD Time Steps (sec)
0
100
200
300
400
500
600
700
800
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
PE Grid

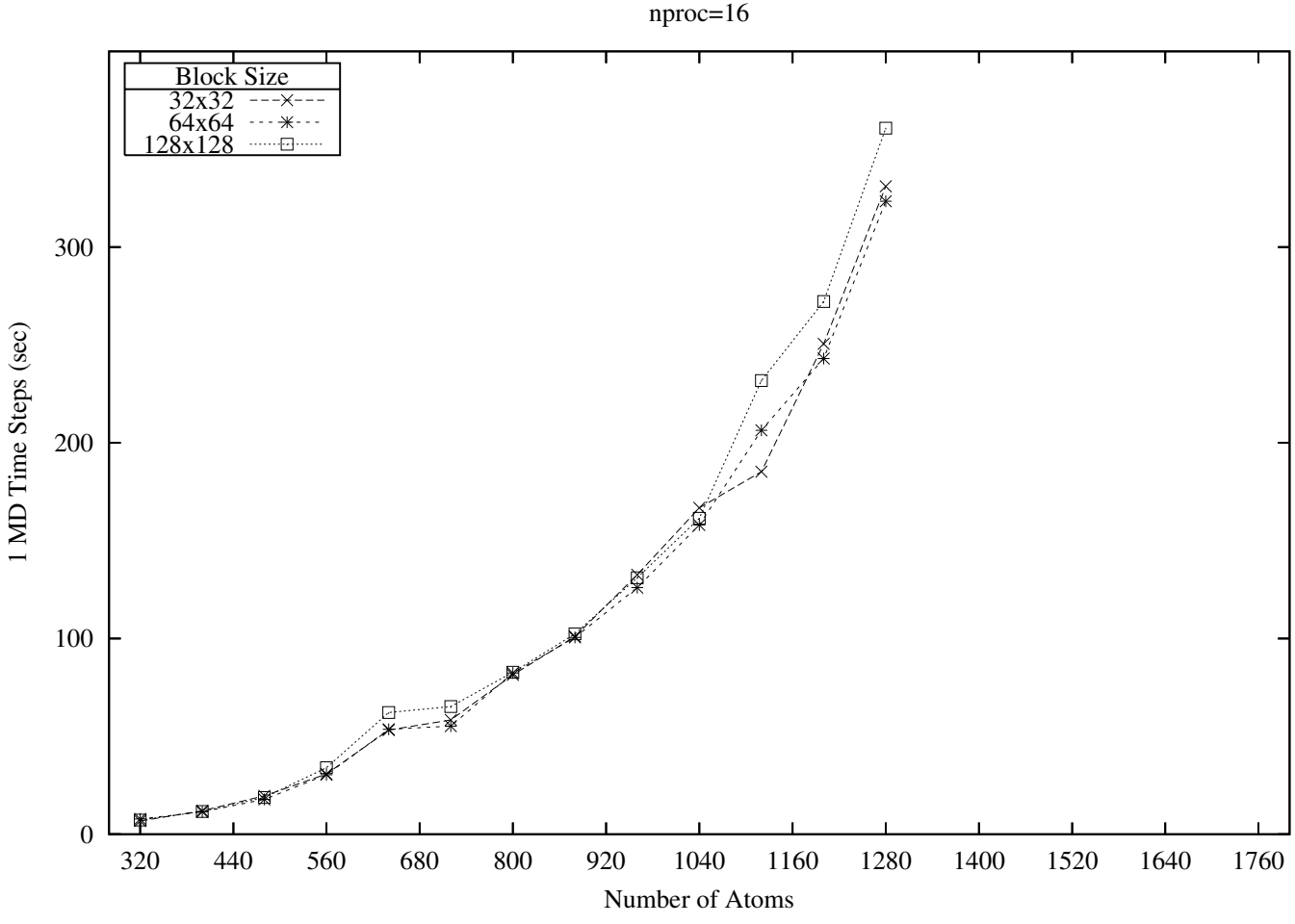
Results 16


160 Atoms
Block Size
2x2
4x4
8x8
16x16
32x32
64x64
128x128
10 MD Time Steps (sec)
PE Grid
0
10
20
30
40
50
60
70
80
90
100
110
120
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

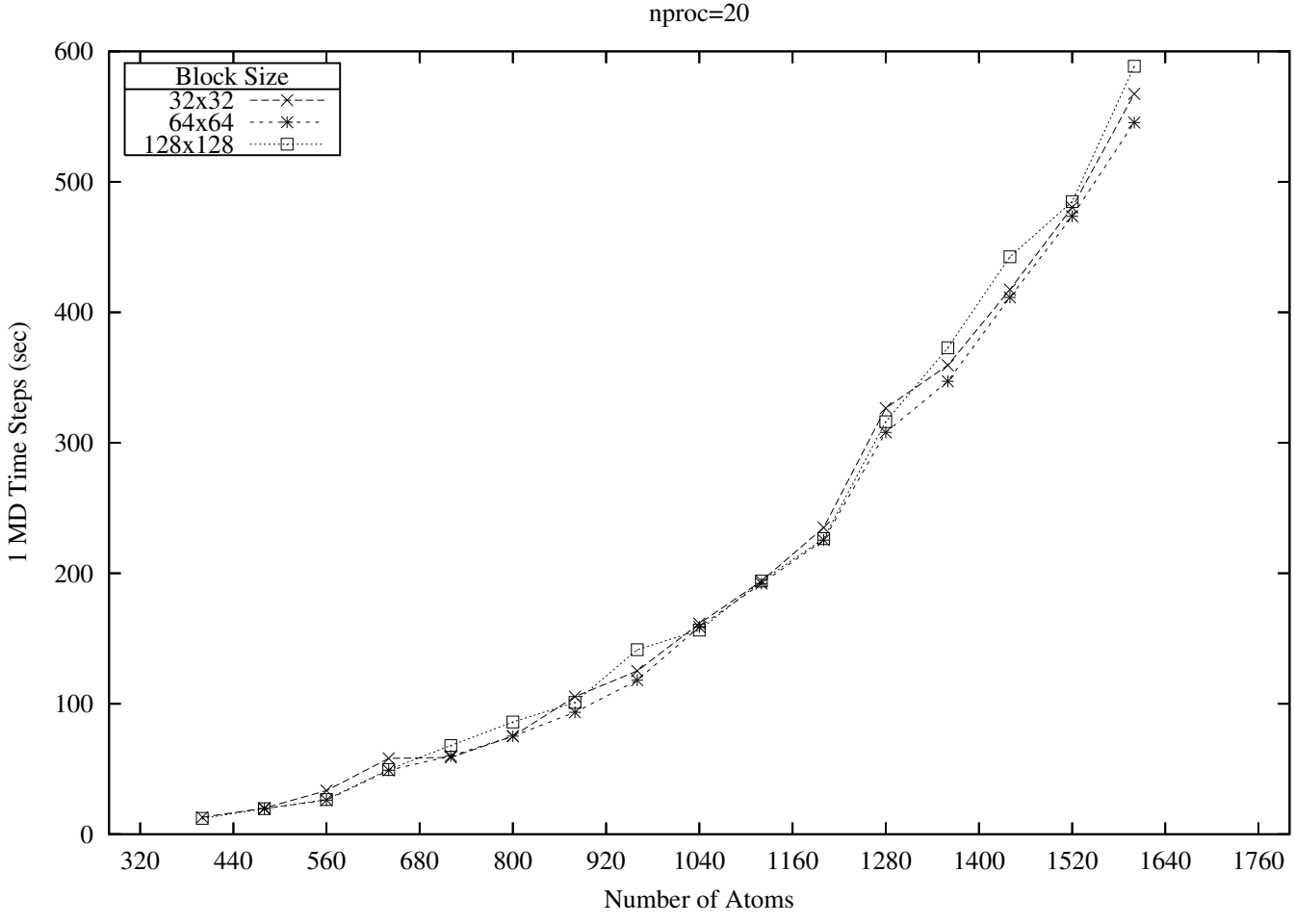
Results 17


4x4 PE Grid
Block Size
2x2
4x4
8x8
16x16
32x32
64x64
128x128
10 MD Time Steps (sec)
0
10
20
30
40
50
60
70
80
90
100
110
120
160
200
240
280
320
360
400
Number of Atoms

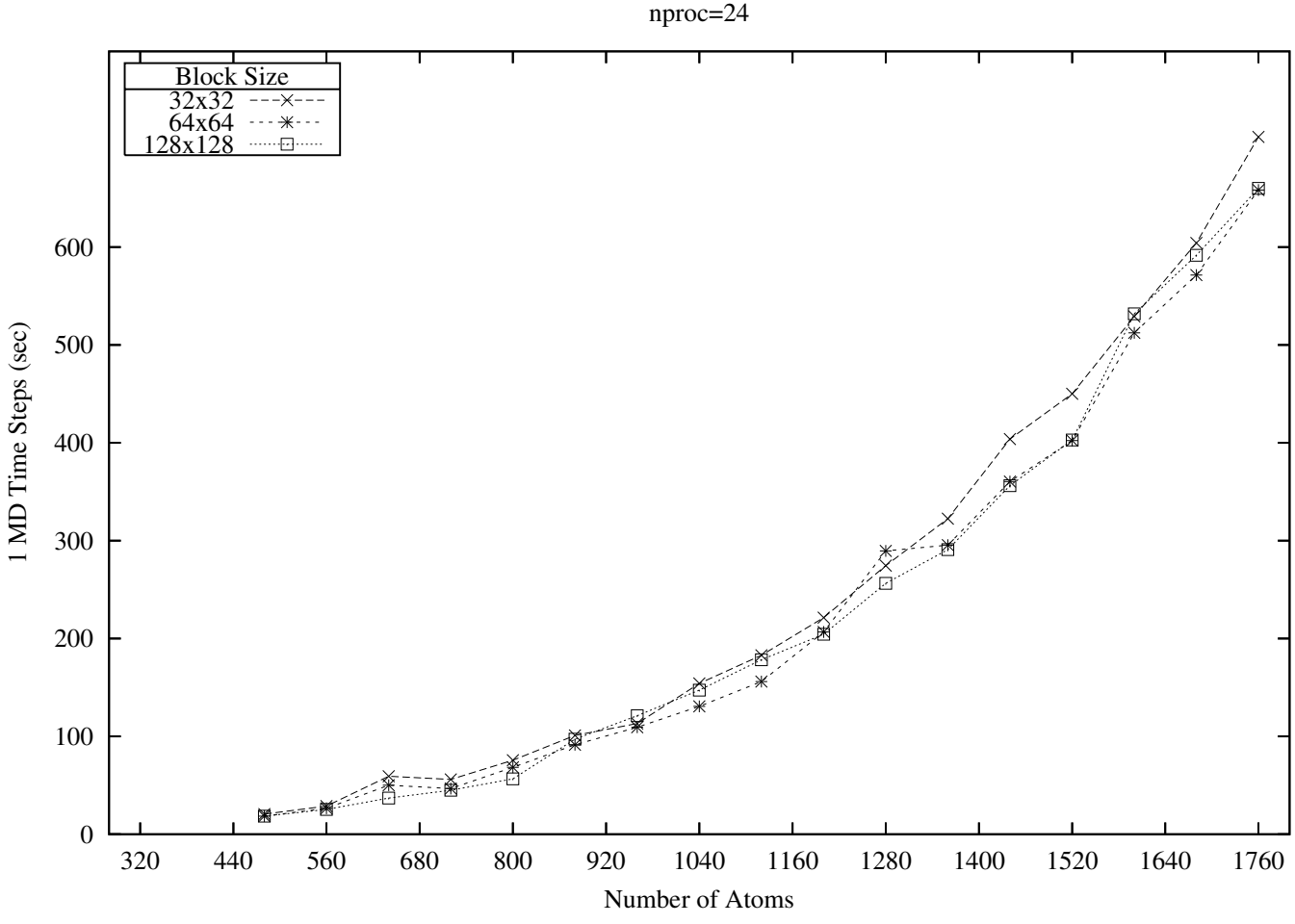
Results 18


nproc=16
Block Size
32x32
64x64
128x128
1 MD Time Steps (sec)
0
100
200
300
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760
Number of Atoms

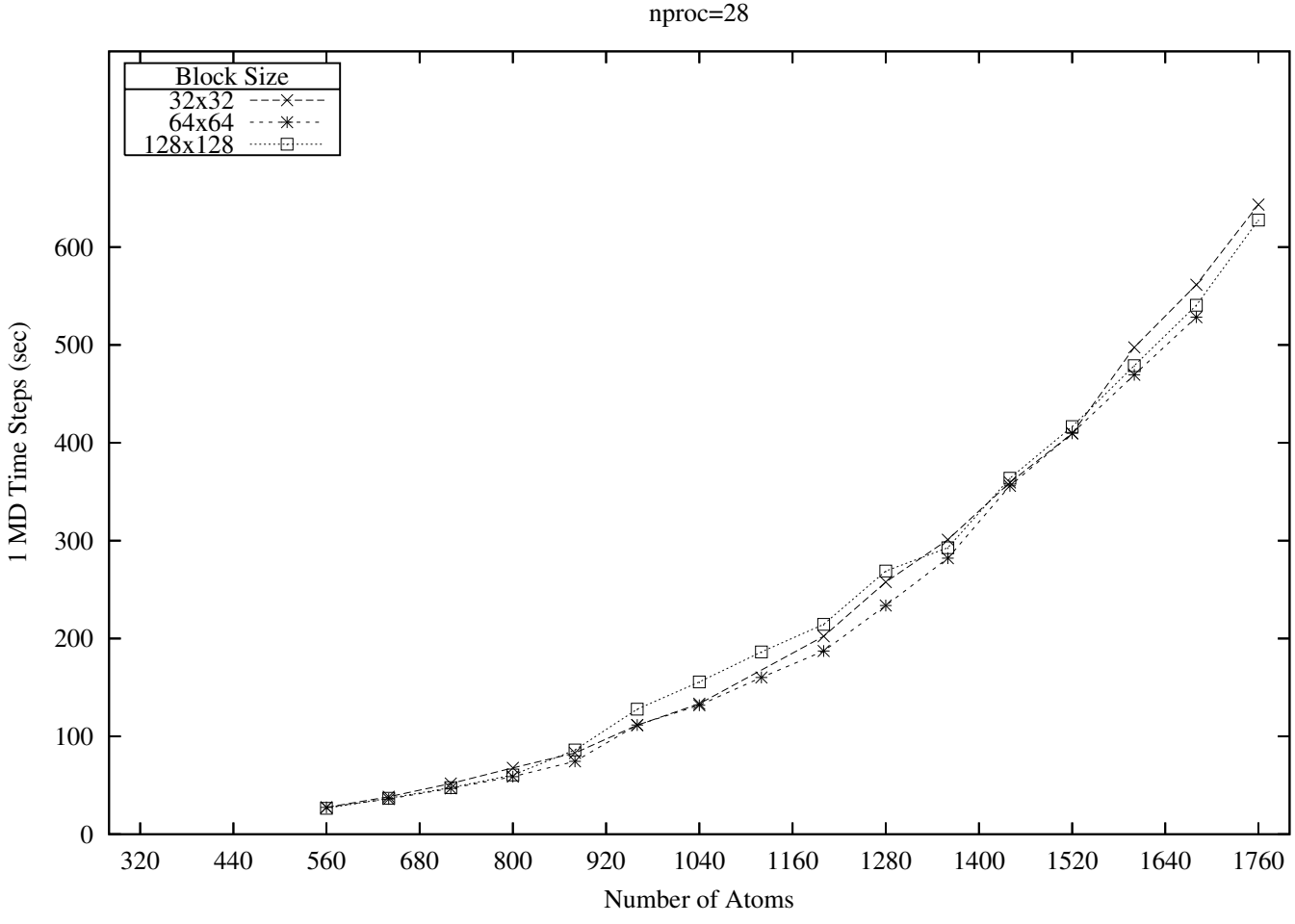
Results 19


nproc=20
Block Size
32x32
64x64
128x128
1 MD Time Steps (sec)
Number of Atoms
0
100
200
300
400
500
600
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760

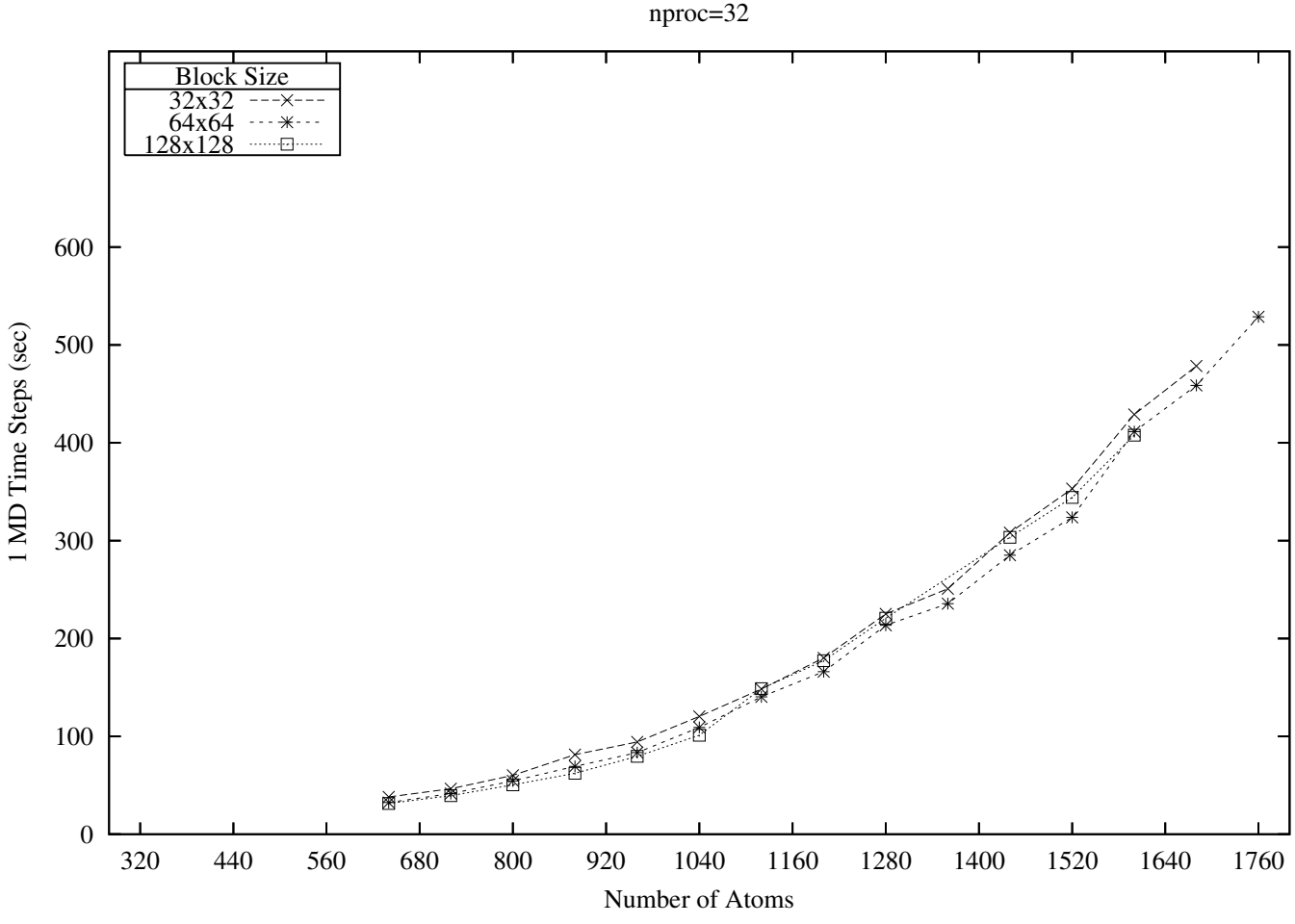
Results 20


nproc=24
Block Size
32x32
64x64
128x128
1 MD Time Steps (sec)
0
100
200
300
400
500
600
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760
Number of Atoms

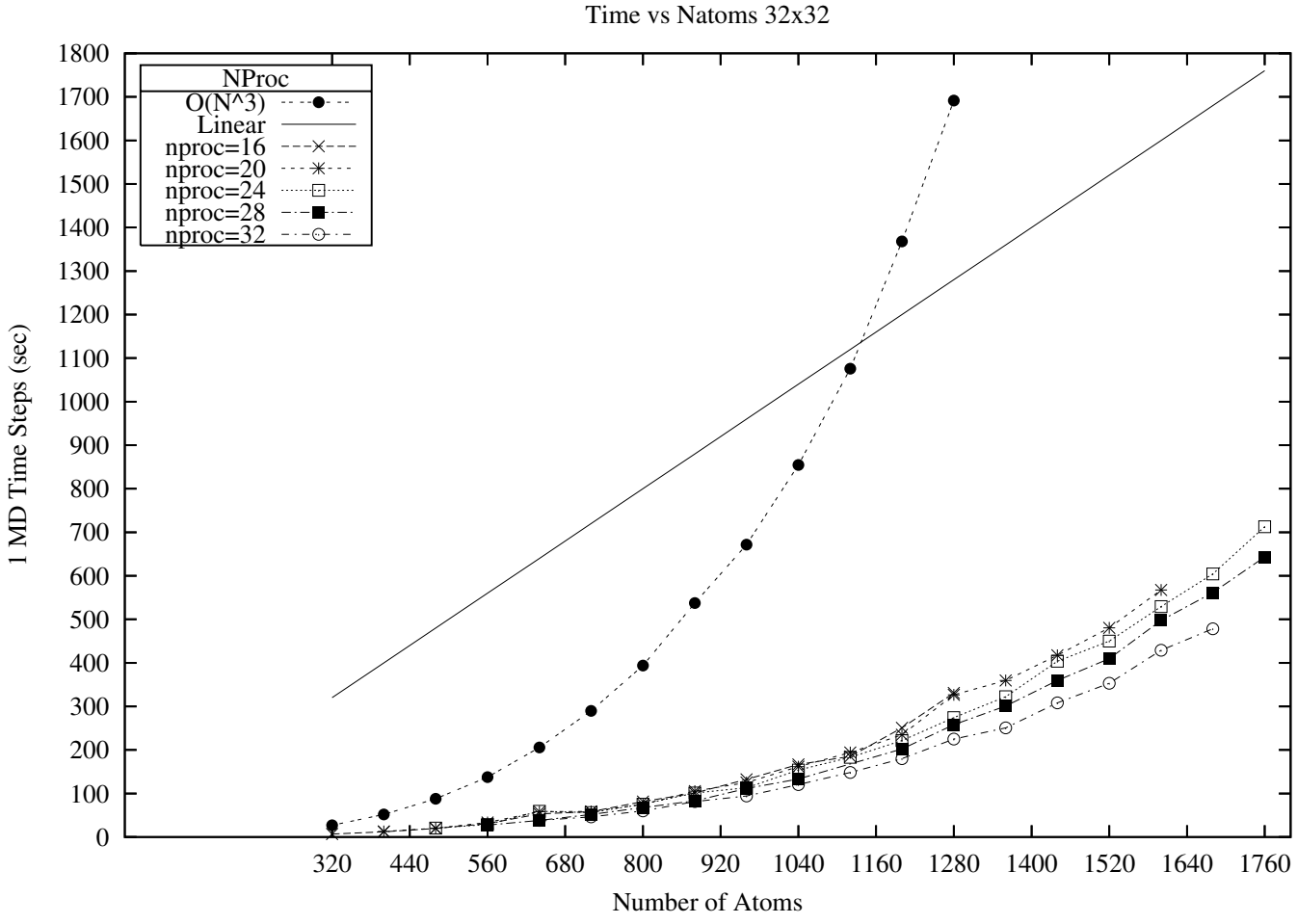
Results 21


nproc=28
Block Size
32x32
64x64
128x128
1 MD Time Steps (sec)
0
100
200
300
400
500
600
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760
Number of Atoms

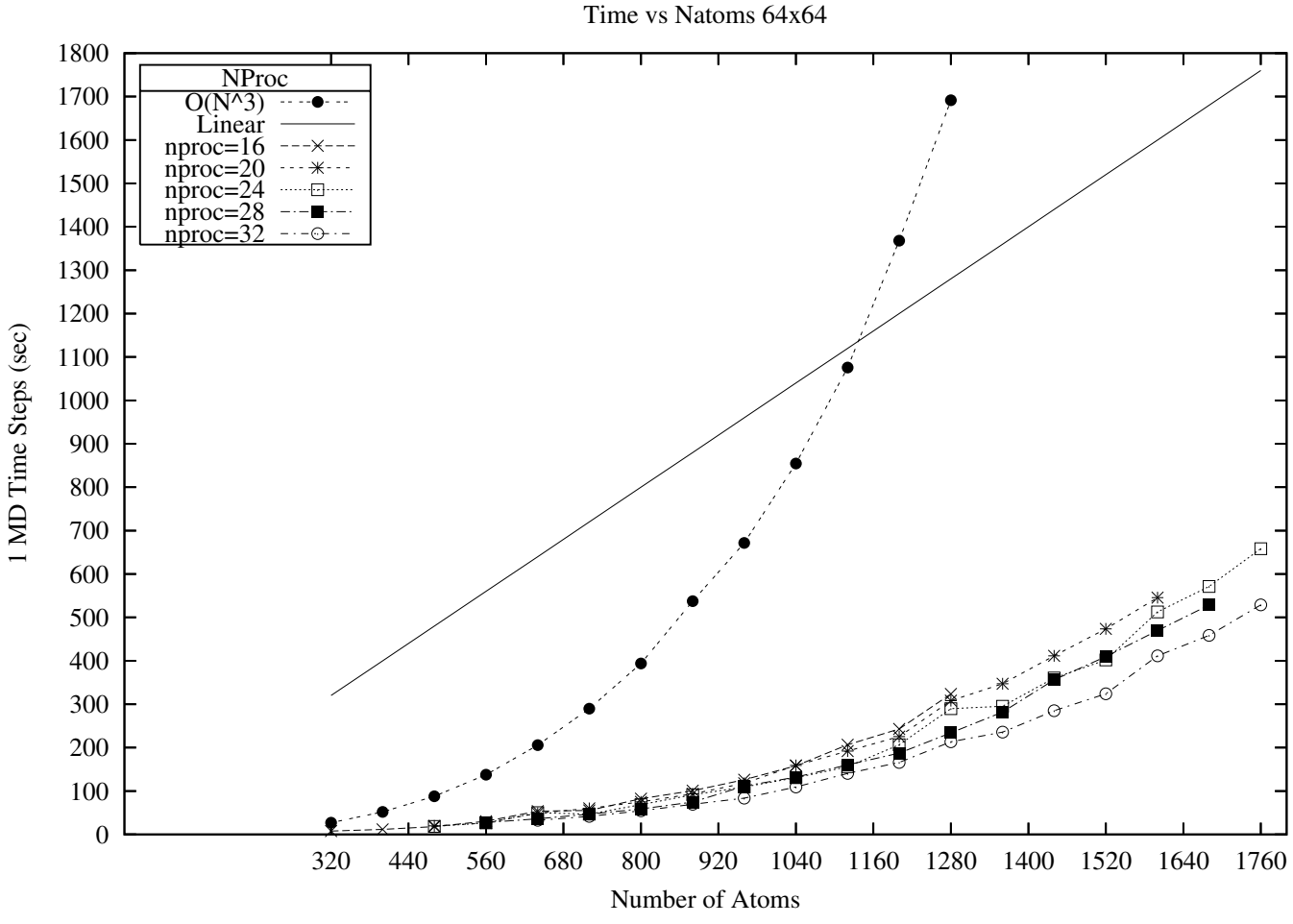
Results 22


nproc=32
Block Size
32x32
64x64
128x128
1 MD Time Steps (sec)
0
100
200
300
400
500
600
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760
Number of Atoms

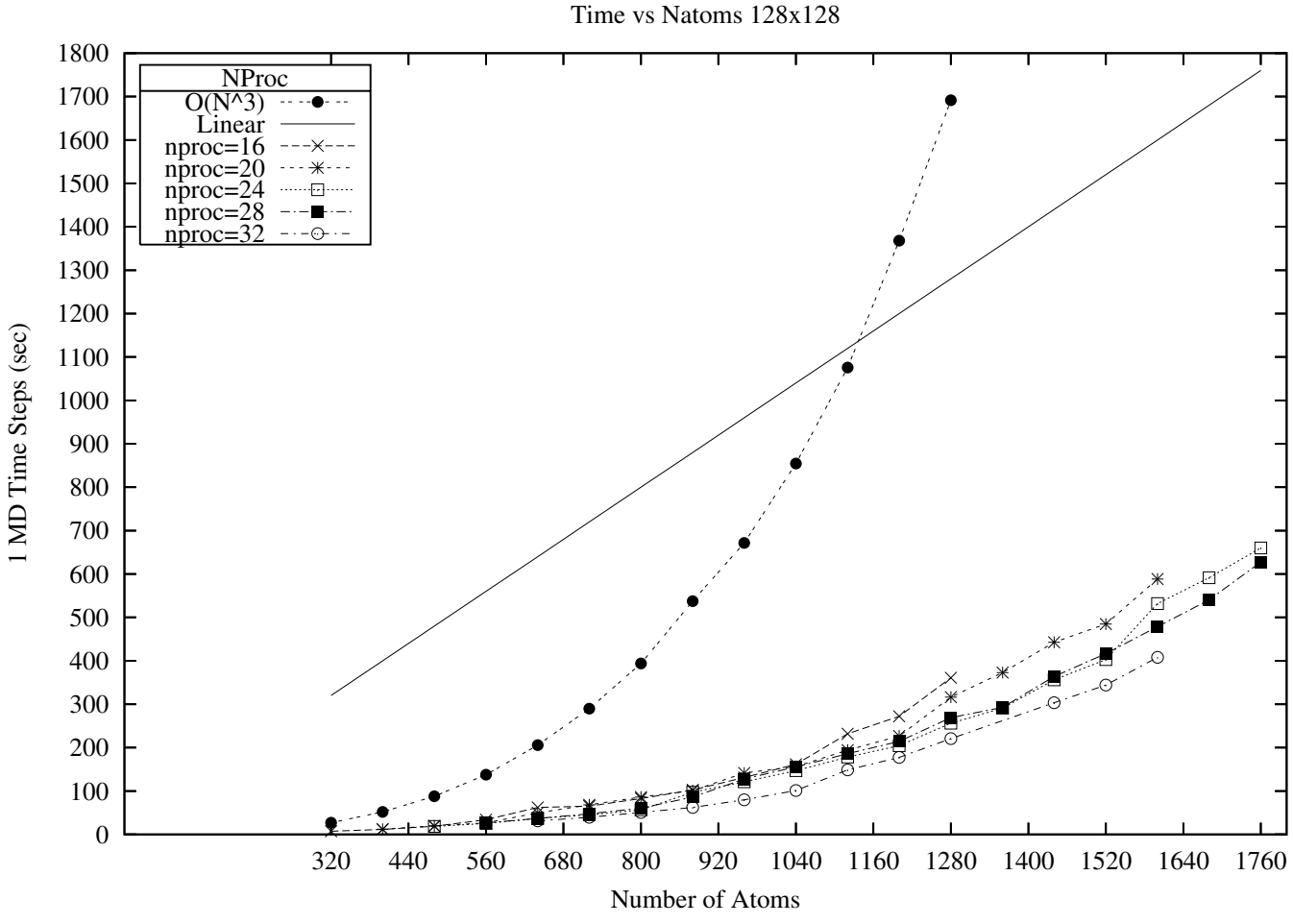
Results 23


Time vs Natoms 32x32
NProc
O(N^3)
Linear
nproc=16
nproc=20
nproc=24
nproc=28
nproc=32
1 MD Time Steps (sec)
Number of Atoms
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760

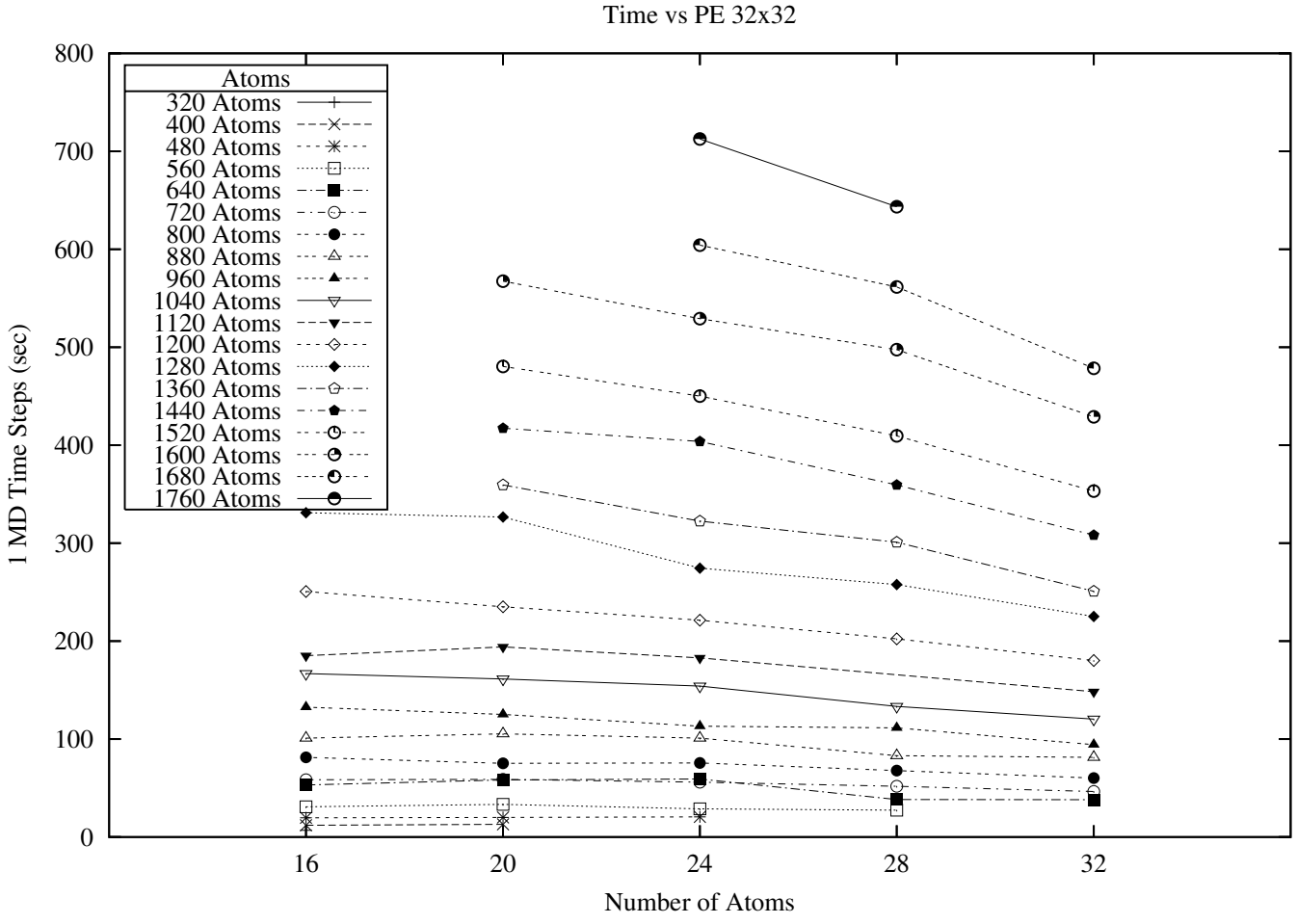
Results 24


Time vs Natoms 64x64
NProc
O(N^3)
Linear
nproc=16
nproc=20
nproc=24
nproc=28
nproc=32
1 MD Time Steps (sec)
Number of Atoms
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760

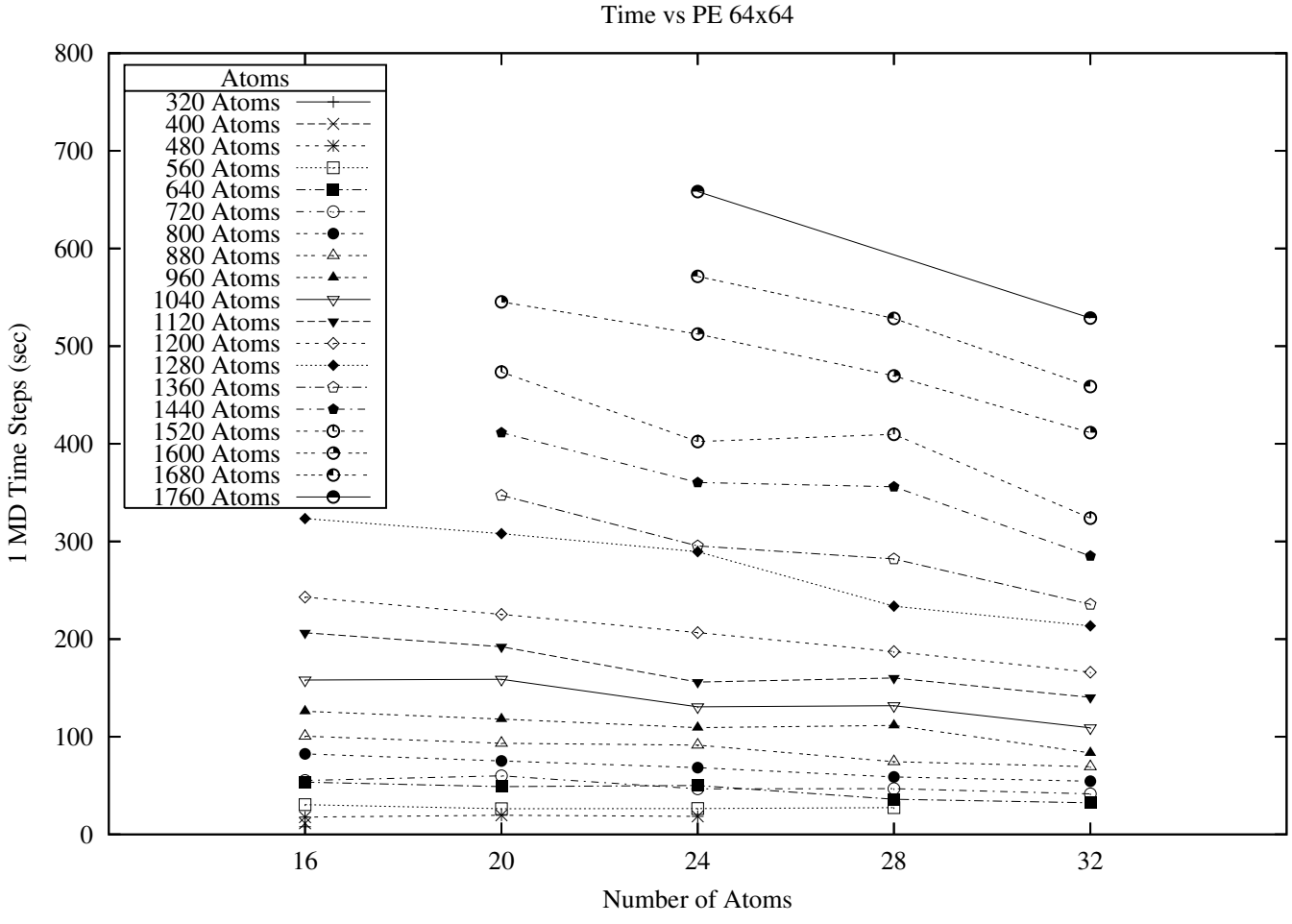
Results 25


Time vs Natoms 128x128
NProc
O(N^3)
Linear
nproc=16
nproc=20
nproc=24
nproc=28
nproc=32
1 MD Time Steps (sec)
Number of Atoms
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760

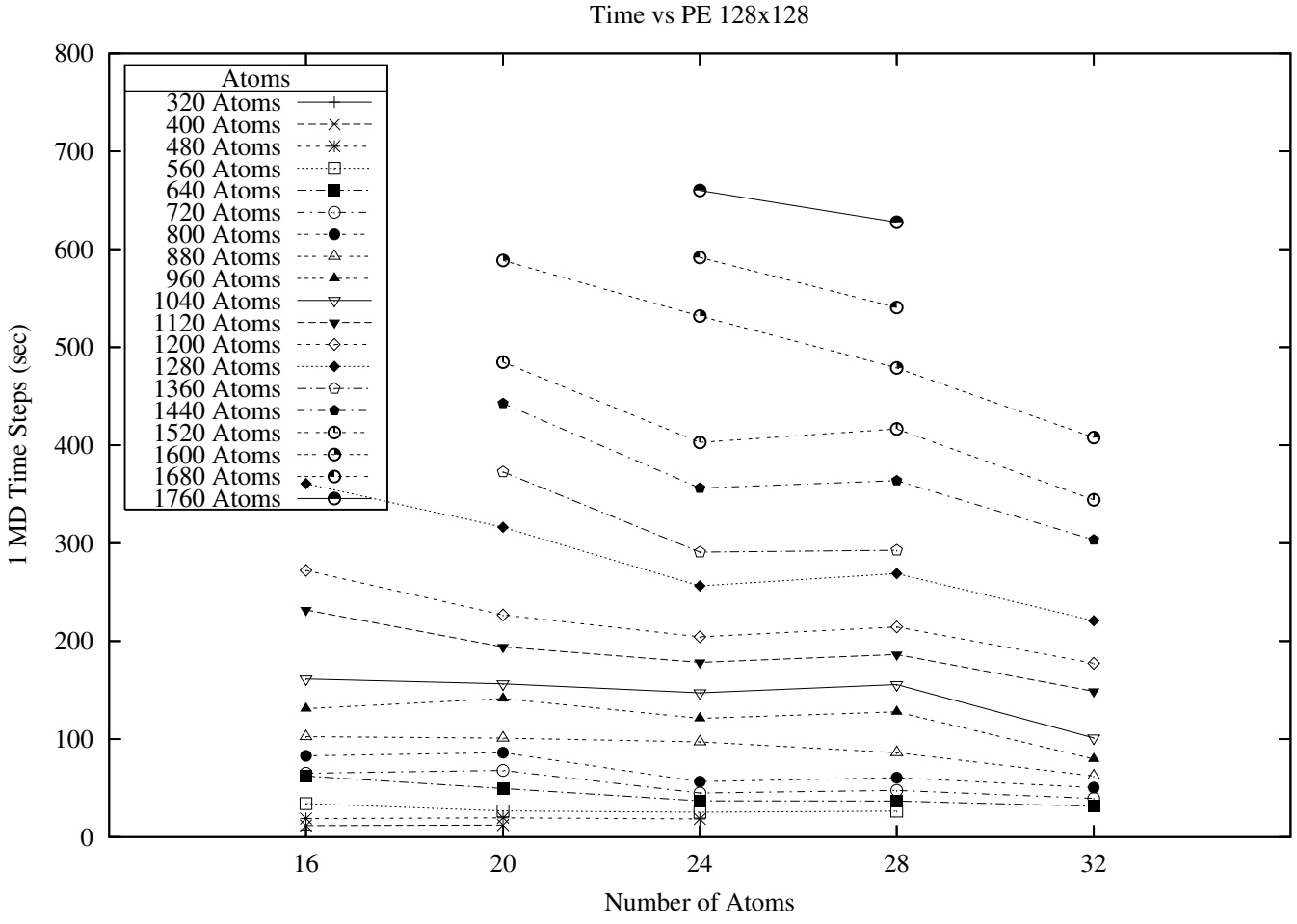
Results 26


Time vs PE 32x32
1 MD Time Steps (sec)
Number of Atoms
0
100
200
300
400
500
600
700
800
16
20
24
28
32
Atoms
320 Atoms
400 Atoms
480 Atoms
560 Atoms
640 Atoms
720 Atoms
800 Atoms
880 Atoms
960 Atoms
1040 Atoms
1120 Atoms
1200 Atoms
1280 Atoms
1360 Atoms
1440 Atoms
1520 Atoms
1600 Atoms
1680 Atoms
1760 Atoms

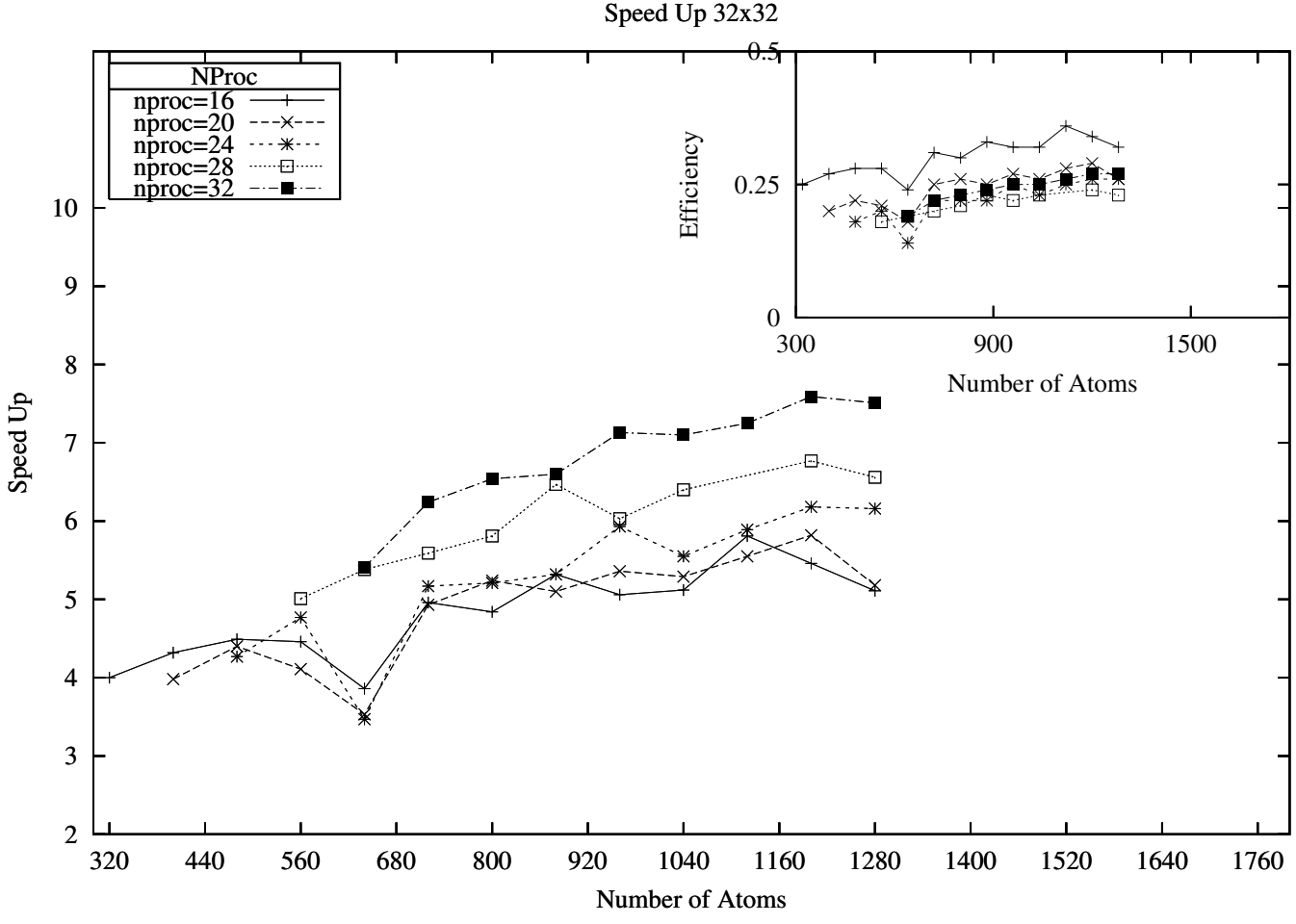
Results 27


Time vs PE 64x64
1 MD Time Steps (sec)
Number of Atoms
0
100
200
300
400
500
600
700
800
16
20
24
28
32
Atoms
320 Atoms
400 Atoms
480 Atoms
560 Atoms
640 Atoms
720 Atoms
800 Atoms
880 Atoms
960 Atoms
1040 Atoms
1120 Atoms
1200 Atoms
1280 Atoms
1360 Atoms
1440 Atoms
1520 Atoms
1600 Atoms
1680 Atoms
1760 Atoms

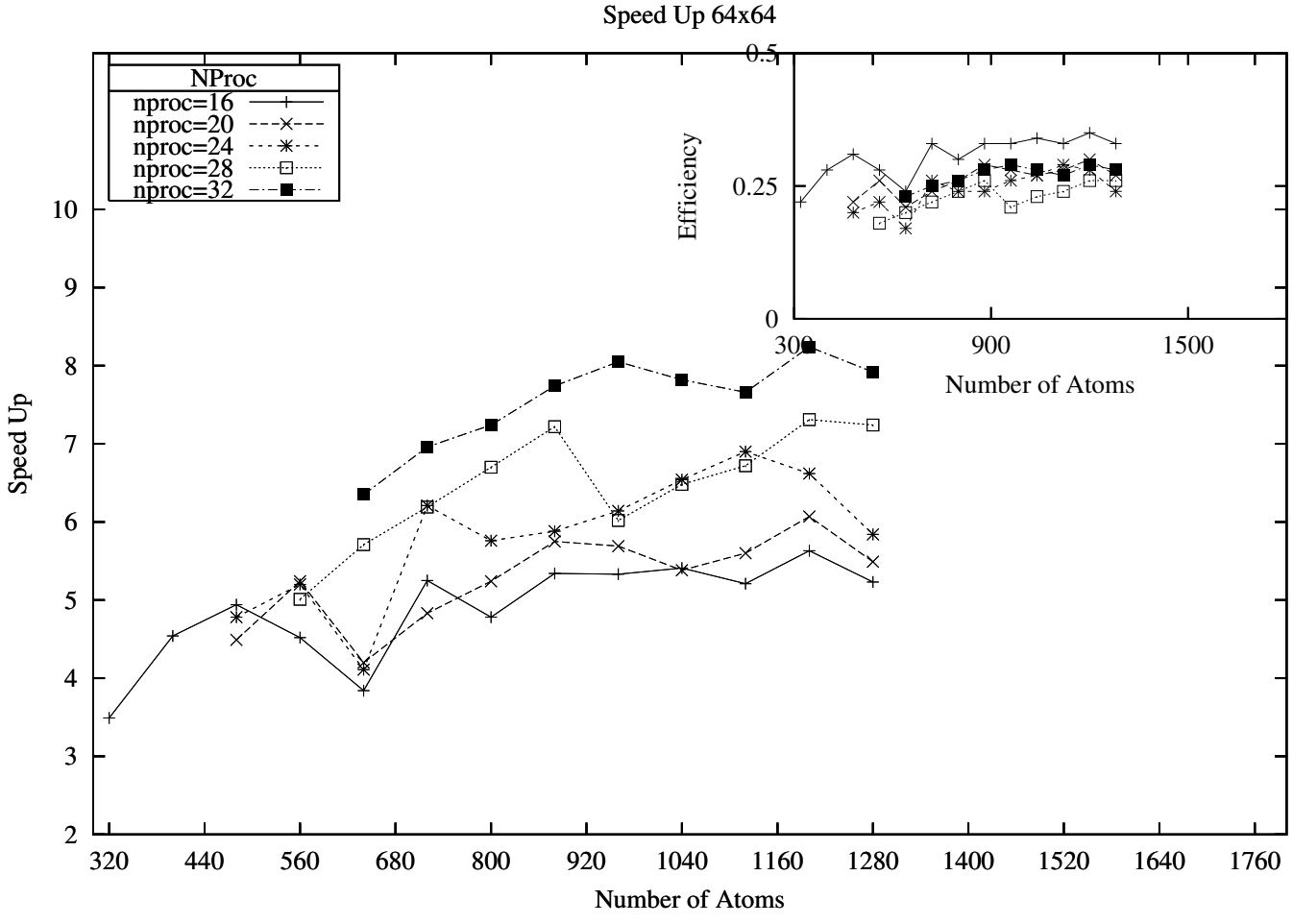
Results 28


Time vs PE 128x128
1 MD Time Steps (sec)
Number of Atoms
0
100
200
300
400
500
600
700
800
16
20
24
28
32
Atoms
320 Atoms
400 Atoms
480 Atoms
560 Atoms
640 Atoms
720 Atoms
800 Atoms
880 Atoms
960 Atoms
1040 Atoms
1120 Atoms
1200 Atoms
1280 Atoms
1360 Atoms
1440 Atoms
1520 Atoms
1600 Atoms
1680 Atoms
1760 Atoms

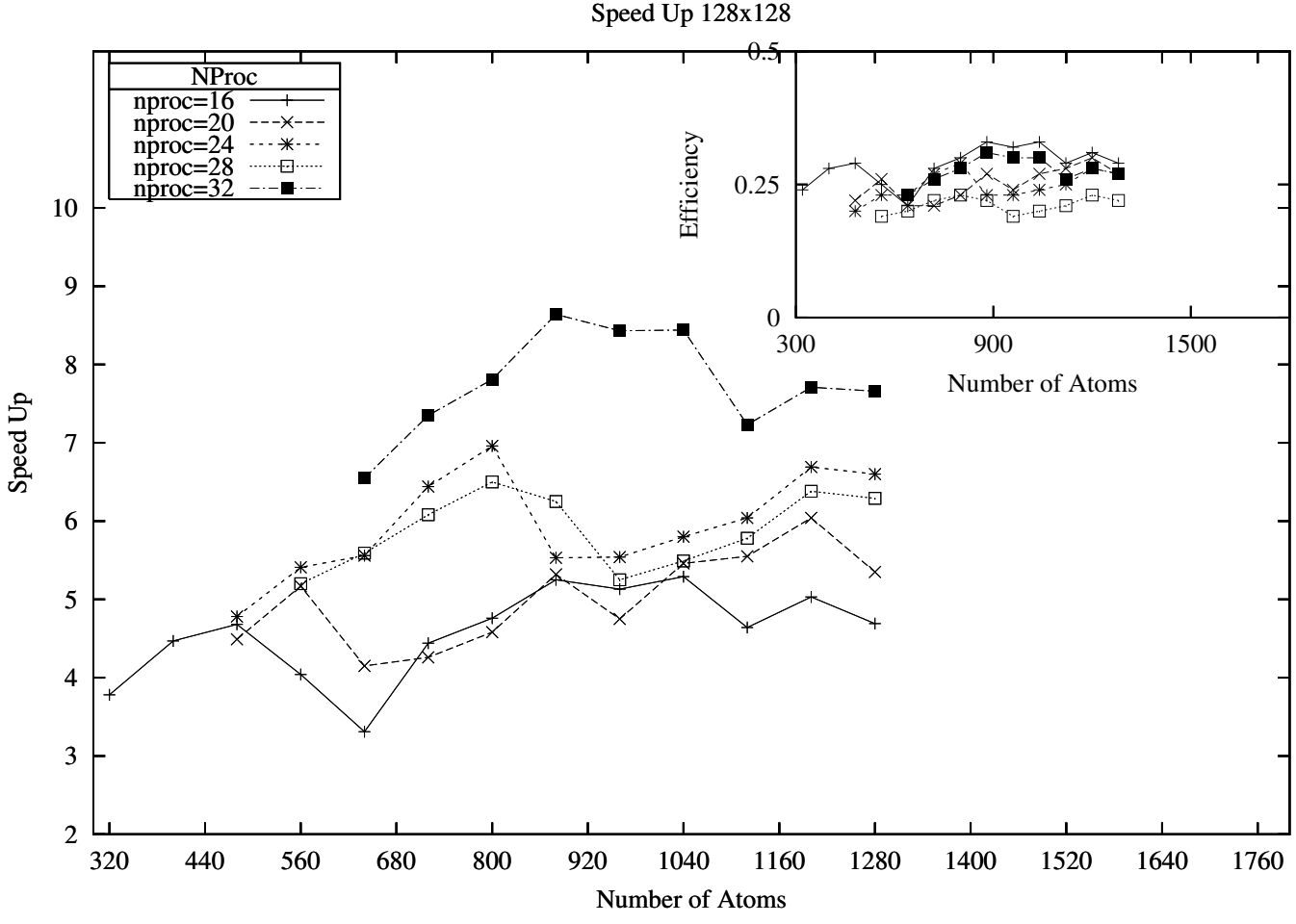
Results 29


Speed Up 32x32
NProc
nproc=16
nproc=20
nproc=24
nproc=28
nproc=32
Speed Up
Number of Atoms
Efficiency
Number of Atoms

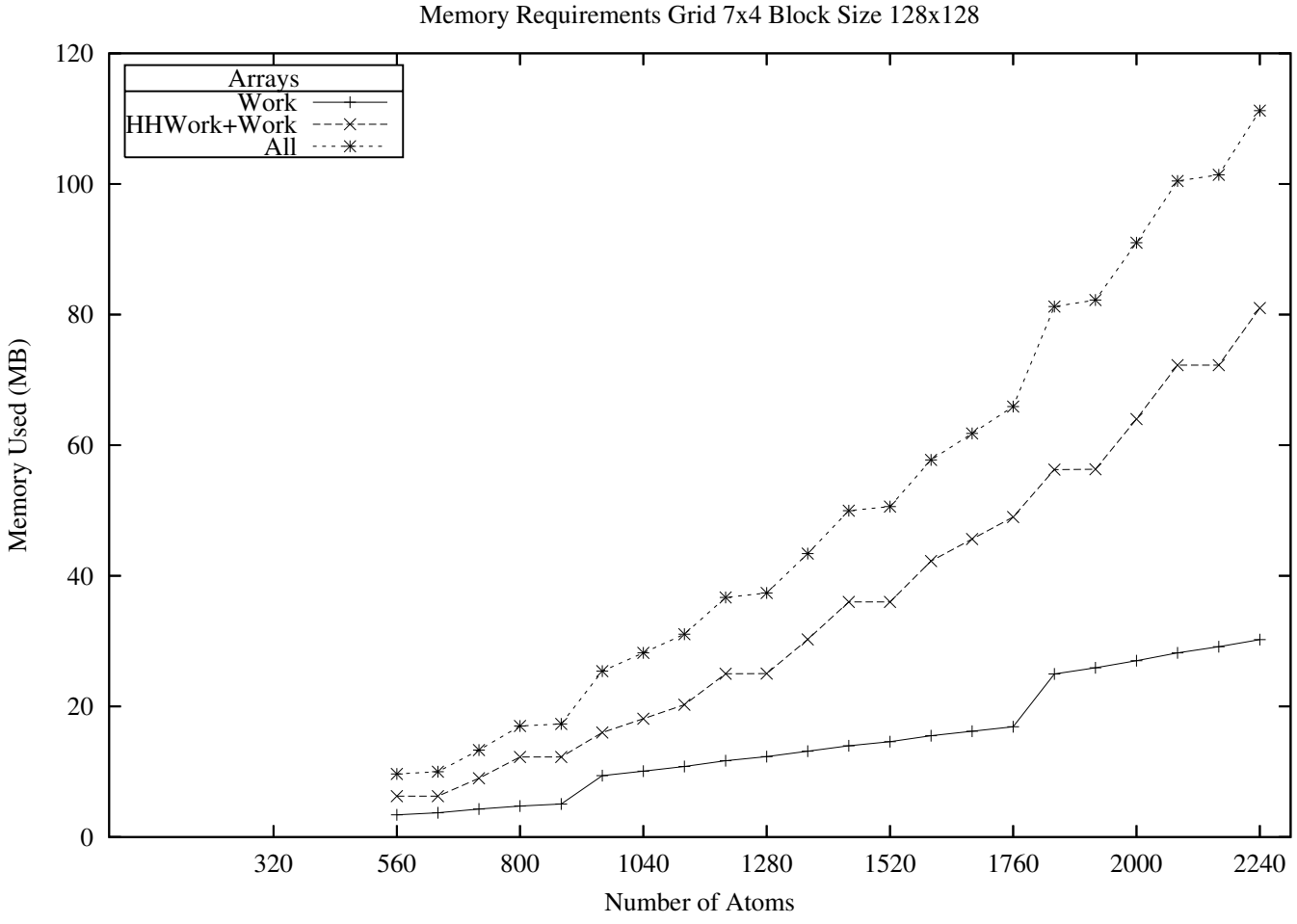
Results 30


Speed Up 64x64
NProc
nproc=16
nproc=20
nproc=24
nproc=28
nproc=32
Speed Up
Number of Atoms
Efficiency
Number of Atoms

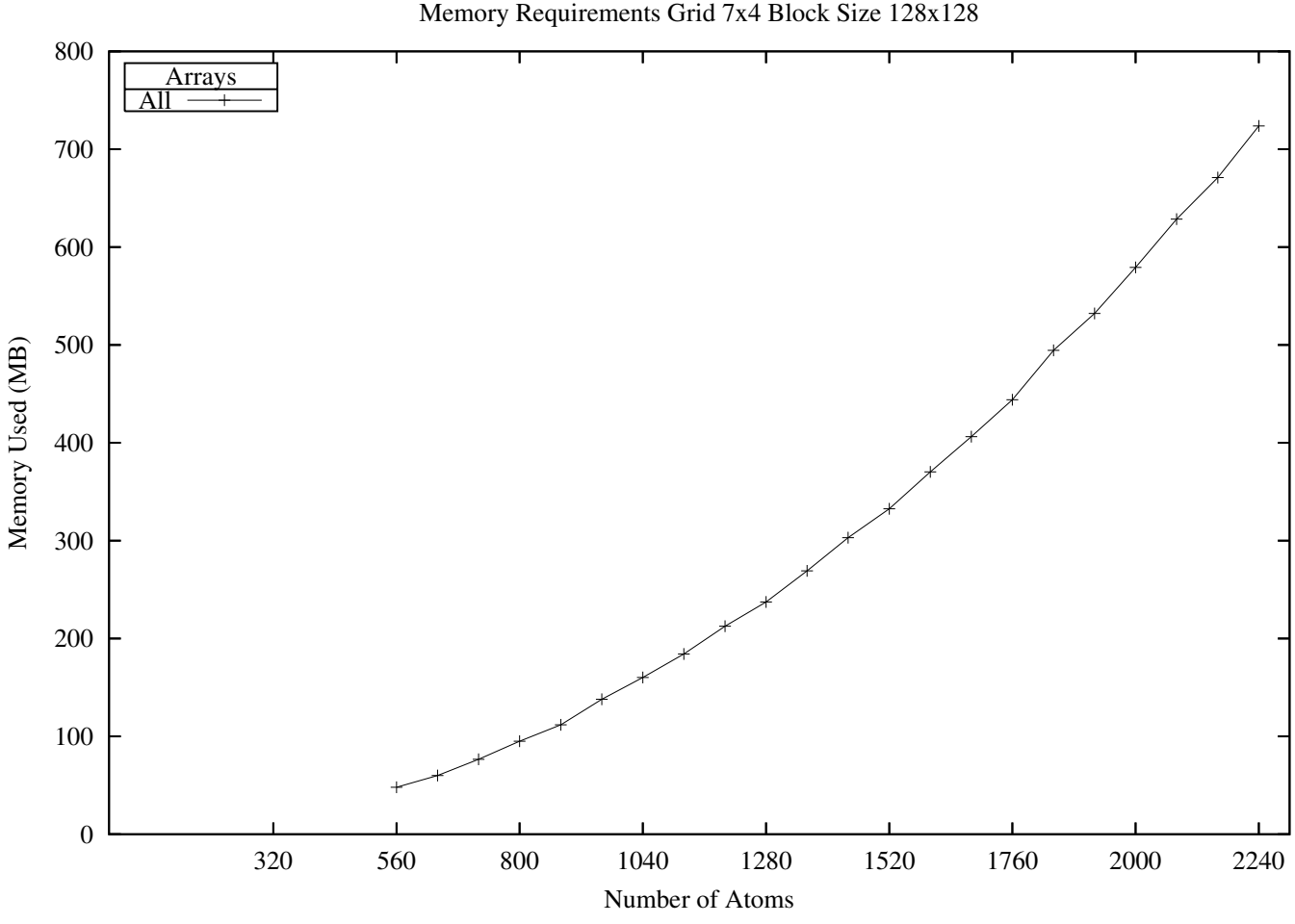
Results 31


Speed Up 128x128
NProc
nproc=16
nproc=20
nproc=24
nproc=28
nproc=32
Speed Up
Number of Atoms
Efficiency
Number of Atoms
2
3
4
5
6
7
8
9
10
320
440
560
680
800
920
1040
1160
1280
1400
1520
1640
1760
0
0.25
0.5
300
900
1500

Results 32


Memory Requirements Grid 7x4 Block Size 128x128
Arrays
Work
HHWork+Work
All
Memory Used (MB)
Number of Atoms
0
20
40
60
80
100
120
320
560
800
1040
1280
1520
1760
2000
2240

Results 33


Memory Requirements Grid 7x4 Block Size 128x128
Arrays
All
Memory Used (MB)
Number of Atoms
0
100
200
300
400
500
600
700
800
320
560
800
1040
1280
1520
1760
2000
2240

Conclusions

- block size of the block cyclic distribution,

Conclusions

- block size of the block cyclic distribution,
- processors grid shape,

Conclusions

- block size of the block cyclic distribution,
- processors grid shape,
- scaling behavior,

Conclusions

- block size of the block cyclic distribution,
- processors grid shape,
- scaling behavior,
- speed-up and efficiency,

Conclusions

- block size of the block cyclic distribution,
- processors grid shape,
- scaling behavior,
- speed-up and efficiency,
- memory requirements,

Conclusions

- block size of the block cyclic distribution,
- processors grid shape,
- scaling behavior,
- speed-up and efficiency,
- memory requirements,
- studied system sizes.