Computer Physics Communications ••• (••••) •••–•••

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers ☆

Benjamin Lindner [a,b,*], Jeremy C. Smith [b,c,**]

[a] Genome Science & Technology, University of Tennessee Knoxville, F337 Walters Life Science, Knoxville, TN 37996, United States
[b] Center for Molecular Biophysics, University of Tennessee/Oak Ridge National Laboratory, 1 Bethel Valley Road, Bldg 6011, Oak Ridge, TN 37830-6309, United States
[c] Department of Biochemistry and Cellular and Molecular Biology, University of Tennessee, M407 Walters Life Sciences, 1414 Cumberland Avenue, Knoxville, TN 37996, United States

ARTICLE INFO

Article history:
Received 4 October 2011
Received in revised form 25 January 2012
Accepted 2 February 2012
Available online xxxx

Keywords:
X-ray
Neutron
Scattering
Molecular dynamics
Massively parallel

ABSTRACT

Massively parallel computers now permit the molecular dynamics (MD) simulation of multi-million atom systems on time scales up to the microsecond. However, the subsequent analysis of the resulting simulation trajectories has now become a high performance computing problem in itself. Here, we present software for calculating X-ray and neutron scattering intensities from MD simulation data that scales well on massively parallel supercomputers. The calculation and data staging schemes used maximize the degree of parallelism and minimize the IO bandwidth requirements. The strong scaling tested on the Jaguar Petaflop Cray XT5 at Oak Ridge National Laboratory exhibits virtually linear scaling up to 7000 cores for most benchmark systems. Since both MPI and thread parallelism is supported, the software is flexible enough to cover scaling demands for different types of scattering calculations. The result is a high performance tool capable of unifying large-scale supercomputing and a wide variety of neutron/synchrotron technology.

Program summary

Program title: Sassena
Catalogue identifier: AELW_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AELW_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: GNU General Public License, version 3
No. of lines in distributed program, including test data, etc.: 1 003 742
No. of bytes in distributed program, including test data, etc.: 798
Distribution format: tar.gz
Programming language: C++, OpenMPI
Computer: Distributed Memory, Cluster of Computers with high performance network, Supercomputer
Operating system: UNIX, LINUX, OSX
Has the code been vectorized or parallelized?: Yes, the code has been parallelized using MPI directives. Tested with up to 7000 processors
RAM: Up to 1 Gbytes/core
Classification: 6.5, 8
External routines: Boost Library, FFTW3, CMAKE, GNU C++ Compiler, OpenMPI, LibXML, LAPACK
Nature of problem: Recent developments in supercomputing allow molecular dynamics simulations to generate large trajectories spanning millions of frames and thousands of atoms. The structural and dynamical analysis of these trajectories requires analysis algorithms which use parallel computation and IO schemes to solve the computational task in a practical amount of time. The particular computational and IO requirements very much depend on the particular analysis algorithm. In scattering calculations a very frequent pattern is that the trajectory data is used multiple times to compute different projections and aggregates this into a single scattering function. Thus, for good performance the trajectory data has to be kept in memory and the parallel computer has to have enough RAM to store a volatile version

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
** Principal corresponding author.
E-mail addresses: lindnerb@ornl.gov (B. Lindner), smithjc@ornl.gov (J.C. Smith).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2012.02.010

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010

of the whole trajectory. In order to achieve high performance and good scalability the mapping of the physical equations to a parallel computer needs to consider data locality and reduce the amount of the inter-node communication.

Solution method: The physical equations for scattering calculations were analyzed and two major calculation schemes were developed to support any type of scattering calculation (all/self). Certain hardware aspects were taken into account, *e.g.* high performance computing clusters and supercomputers usually feature a 2 tier network system, with Ethernet providing the file storage and infiniband the inter-node communication via MPI calls. The time spent loading the trajectory data into memory is minimized by letting each core only read the trajectory data it requires. The performance of inter-node communication is maximized by exclusively utilizing the appropriate MPI calls to exchange the necessary data, resulting in an excellent scalability. The partitioning scheme developed to map the calculation onto a parallel computer covers a wide variety of use cases without negatively effecting the achieved performance. This is done through a 2D partitioning scheme where independent scattering vectors are assigned to independent parallel partitions and all communication is local to the partition.

Additional comments: !!!!! The distribution file for this program is approximately 36 Mbytes and therefore is not delivered directly when download or E-mail is requested. Instead an html file giving details of how the program can be obtained is sent. !!!!!

Running time: Usual runtime spans from 1 min on 20 nodes to 2 h on 2000 nodes. That is 0.5–4000 CPU hours per execution.

1. Introduction

The use of analytical theory to interpret spectroscopic or scattering experiments performed on complex molecular systems often suffers from underdetermination. One example of this is the inverse scattering problem in small angle X-ray or neutron scattering (SAS) theory in which various shapes of a scattering particle often can yield close to the same spherically-averaged scattering pattern [1]. Another example is in dynamic neutron scattering in which a relatively simple signal may result from the superposition of many relaxation processes, rendering difficult the assignment of peaks [2].

To overcome the problem of underdetermination computer simulation is often used to bridge theory and experiment. Of particular importance is molecular dynamics (MD) simulation involving stepwise integration of Newton's equation and building up atomic trajectories in space and time. For X-ray and neutron scattering experiments MD solves the forward scattering problem [3], producing scattering profiles from the trajectories.

For biological systems several software packages exist which provide solutions to the forward scattering problem from MD for different kinds of scattering experiments. Examples of these are SASSIM [4,5], which computes a time-averaged small angle scattering diagram for a macromolecule in solution, SERENA [6–8], which computes X-ray diffuse scattering and nMoldyn [9], which calculates incoherent and coherent neutron scattering spectra from MD simulations. Many other software packages exist that do not necessarily use MD, such as CRYSOL [10], which fits experimental SAS data with various representations of the target.

With the establishment of petaflop supercomputing facilities, *e.g.*, the Cray XT5 Jaguar machine at Oak Ridge National Laboratory, MD code can now simulate multi-million atom systems on biologically-relevant timescales [11]. These simulations produce trajectories of sizes and lengths such that their analysis may itself require parallel computation of considerable size. X-ray and neutron scattering calculations are an example of highly CPU-intensive analysis. To solve the forward scattering problem for simulations using supercomputers, the corresponding software must scale to a point at which the time to compute drops to a practical level. This paper introduces such a software solution.

Highly scalable data analysis software of the kind presented here must solve two independent problems: avoiding unnecessary IO traffic and efficiently using the parallel bandwidth (CPU and IO). Both these criteria can be met by intelligent data staging, partitioning, and calculation schemes. The performance analysis shows that the present software, Sassena,[1] is able to scale well, thus enabling the calculation of scattering intensities from MD with a fidelity previously unattainable. The utility of the software is illustrated by two scattering functions calculated from the MD simulation of a cellulose fiber: Wide Angle X-ray Scattering (WAXS) and Inelastic X-ray Scattering (IXS). The former illustrates how scattering functions can probe crystalline fingerprints, while the latter illustrates their use for determining phonon dispersion relations. The present software takes a necessary step towards unification of the use of extreme-scale supercomputing and neutron/X-ray scattering facilities.

2. Theory

The theory of X-ray and neutron scattering is well known [12–14]. In the classical description of scattering theory the atomic particles are sources of plane waves, which may interfere in different ways dependent on the type of scattering experiment. The scattering patterns arise from the constructive interference of atomic scattering amplitudes $a(\vec{q},t)$:

$$a_n(\vec{q},t) = b_n(\vec{q}) \cdot e^{i\cdot\vec{q}\cdot\vec{r}_n(t)} \tag{1}$$

where $\vec{q}$ is the scattering vector, $b_n(\vec{q})$ is the atomic prefactor, and $\vec{r}_n(t)$ is the time dependent Cartesian position vector of atom n. The total scattering amplitude for all atoms, $A(\vec{q},t)$, and the associated scattering intensity, $F(\vec{q},t)$, is given by:

$$F(\vec{q},t) = A(\vec{q},t)\cdot A^*(\vec{q},t) = \left|A(\vec{q},t)\right|^2 \tag{2}$$

$$\text{where } A(\vec{q},t) = \sum_n a_n(\vec{q},t) \tag{3}$$

and $A^*(\vec{q},t)$ is the complex conjugate of $A(\vec{q},t)$. $A(\vec{q},t)$ may be associated with a higher organizational structure in which the individual atoms are incorporated.

The atomic prefactor, b_n, is different for X-ray and neutron scattering. In X-ray scattering it represents the form factor of the electronic shell of the respective atom and is approximated as a series of Gaussians [15]:

$$b_n(\vec{q}) = \sum_i c_i \cdot e^{-d_i\cdot|\vec{q}|^2} \tag{4}$$

[1] The name Sassena was derived from SASS-IM and SER-ENA: SASS-ENA.

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010

where the set of c_i and d_i are empirically derived constants. In neutron scattering b_n is the atomic scattering length, which is different for each isotope and independent of $|\vec{q}|$. The variation of the atomic scattering length due to isotopic distribution and spin orientation gives rise to coherent and incoherent scattering, which is described by two distinct scattering lengths for each isotope type: b^{coh} and b^{inc} [16]. The resulting total scattering function for neutron scattering is therefore split into a coherent part, F_{coh}, and an incoherent part, F_{inc}:

$$F(\vec{q},t) = F_{coh}(\vec{q},t) + F_{inc}(\vec{q},t) \tag{5}$$

$$F_{coh}(\vec{q},t) = \left|\sum_n a_n^{coh}(\vec{q},t)\right|^2 \tag{6}$$

$$F_{inc}(\vec{q},t) = \sum_n \left|a_n^{inc}(q,\vec{t})\right|^2 = \sum_n \left(b_n^{inc}\right)^2 \tag{7}$$

The coherent scattering function is based on the total scattering amplitude $A(\vec{q},t)$ and uses b^{coh}, while the incoherent function is associated with atomic scattering amplitudes $a_n(\vec{q},t)$ and uses b^{inc}.

In dynamic scattering experiments the scattering amplitudes at times t and $t+\tau$ are superimposed, resulting in a correlated time signal of the form:

$$F(\vec{q},t,\tau) = A(\vec{q},t)\cdot A^*(\vec{q},t+\tau) \tag{8}$$

or

$$\sum_n f_n(\vec{q},t,\tau) = \sum_n a_n(\vec{q},t)\cdot a_n^*(\vec{q},t+\tau) \tag{9}$$

where $F(\vec{q},t,\tau)$ and $f_n(\vec{q},t,\tau)$ are the total and atom-correlated time signals, respectively.

2.1. Solution scattering

In solution scattering a large number of identical solute particles are in a solvent permitting solute rotational and translational motion. Thus, there is no preferred orientation and the scattering signal becomes isotropic, *i.e.*, independent of the sample orientation: $F(\vec{q},t) \to F(q,t)$. Also, the system can be in any of its thermodynamically allowed states, causing the scattering to represent the ensemble average of the individual particle dynamics, which is taken as the MD time average: $F(q,t) \to \langle F(q,t)\rangle_t$. The scattering expression associated with solution scattering conditions is therefore:

$$F(q) = N_P\left\langle\left\langle F(\vec{q},t)\right\rangle_t\right\rangle_\Omega \tag{10}$$

where N_P is the number of scattering particles and $\langle\rangle_\Omega$ performs the orientational averaging for all orientations of the scattering vector $\vec{q}$. For solution scattering, the solute atomic scattering factors can be adjusted to represent the proper scattering length contrast, as is outlined in Ref. [4].

2.2. Crystal diffraction

In a crystal diffraction experiment the scattering particles are orientationally aligned. The resulting scattering pattern is a convolution of the scattering due to the lattice and the scattering unit. The scattering expression associated with diffraction is:

$$F(\vec{q}) = N_P\left\langle F(\vec{q},t)\right\rangle_t \tag{11}$$

where N_P is the number of scattering particles. The scattering function due to the crystal lattice is neglected for simplicity, but can be modeled explicitly in MD simulations.

2.3. Dynamic scattering

Dynamic scattering experiments allow the measurement of either the intermediate scattering function, $I(\vec{q},\tau)$, or the dynamic structure factor, $S(\vec{q},\omega)$. Since $S(\vec{q},\omega)$ is simply the Fourier transform of $I(\vec{q},\tau)$, only the intermediate scattering function, $I(\vec{q},\tau)$, is discussed in the following. In dynamic X-ray and coherent neutron scattering the scattering intensities arise from a superposition of the total scattering amplitude, $A(\vec{q},t)$, at two times t and $t+\tau$, given by:

$$I_{coh}(q,\tau) = \left\langle\left\langle F(\vec{q},t,\tau)\right\rangle_t\right\rangle_\Omega \tag{12}$$

which probes structural–temporal correlations. Dynamic incoherent neutron scattering measures the superposition of the individual atomic scattering amplitudes, $a_n(\vec{q},t)$, at t and $t+\tau$, and is given by

$$I_{inc}(q,\tau) = \sum_n \left\langle\left\langle f_n(\vec{q},t,\tau)\right\rangle_t\right\rangle_\Omega \tag{13}$$

which probes correlations in the displacements of individual atoms. For large correlation times, *i.e.* $\tau \to \infty$, the coherent and incoherent dynamic scattering function converge, allowing the function to be split into time-dependent and -independent parts:

$$I_{coh,inc}(q,\tau) = I^*_{coh,inc}(q,\tau) + I_{coh,inc}(q,\infty) \tag{14}$$

where $I_{coh,inc}(q,\infty)$ are the elastic coherent and incoherent structure factors (EISF) and can be approximated without the need to compute the autocorrelation:

$$I_{coh}(q,\infty) = \left\langle\left|\left\langle A(\vec{q},t)\right\rangle_t\right|^2\right\rangle_\Omega \tag{15}$$

$$I_{inc}(q,\infty) = EISF(q) = \sum_n \left\langle\left|\left\langle a_n(\vec{q},t)\right\rangle_t\right|^2\right\rangle_\Omega \tag{16}$$

3. Software and design

The software was designed to calculate the scattering functions outlined in the above theory section. The computed scattering function is determined by two independent settings: Scattering type (TYPE) and digital signal processing procedure (DSP). The TYPE can be either *all* or *self* and is used to distinguish between coherent and incoherent scattering, respectively. The DSP setting is used to compute either a real-time or an autocorrelated time signal of the scattering amplitude and can be either *plain*, *square*, or *autocorrelate*. The option *plain* is a pass-through option and does not modify the computed scattering amplitudes, which is used here to compute the elastic component of the dynamic scattering function.

The $\vec{q}$ vector space is selected by the user to match the requirements of the corresponding scattering experiment. This includes the setting of the resolution of any orientational averaging. The HDF5 formatted output file contains four distinct datasets: *fqt*, *fq*, *fq2*, *fq0*. *fqt* is the post-processed time-dependent scattering signal, *fq* is the time average of *fqt*, *fq2* is the squared value of *fq* and *fq0* is the zero time component of *fqt*.

Table 1 lists the mapping between the scattering amplitudes and the output fields, taking into account the various settings for TYPE and DSP. The extensive mapping between the computed signal and the final output covers a wide spectrum of potential scattering calculations without requiring any algorithmic changes. Table 2 lists possible settings for the above switches, the associated scattering function, and a potential application. The distinction between X-ray and neutron scattering is implemented as a set of database files and allows the user to easily adjust the atomic scattering factors, b_n, to custom values.

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010

Table 1
(a) Mapping of the computed signal (after the DSP) and the output field. $\langle\rangle_t$ and $\langle\rangle_\Omega$ denote time and orientational averaging, respectively. (b) Mapping between function symbol and scattering function dependent on the DSP setting.

(a)

Output	TYPE setting	
	all	*self*
fqt	$\langle X\rangle_\Omega$	$\sum_n \langle x_n\rangle_\Omega$
fq	$\langle\langle X\rangle_t\rangle_\Omega$	$\sum_n \langle\langle x_n\rangle_t\rangle_\Omega$
fq2	$\langle\lvert\langle X\rangle_t\rvert^2\rangle_\Omega$	$\sum_n \langle\lvert\langle x_n\rangle_t\rvert^2\rangle_\Omega$
fq0	$\langle X(t=0)\rangle_\Omega$	$\sum_n \langle x_n(t=0)\rangle_\Omega$

(b)

Symbol	DSP setting		
	autocorrelate	*square*	*plain*
X	$\langle F(\vec{q},t,\tau)\rangle_t$	$F(\vec{q},t)$	$A(\vec{q},t)$
x_n	$\langle f_n(\vec{q},t,\tau)\rangle_t$	$f_n(\vec{q},t)$	$a_n(\vec{q},t)$

The computed scattering signals are based on the discrete Cartesian coordinates supplied as one of the major trajectory formats for molecular dynamics (XTC, TRR, DCD). A major design decision was to store the data completely into memory before performing any scattering calculations, for several reasons. First, most scattering calculations require repeated computations using the same trajectory data with a synchronization point after each iteration, which means that any benefit of overlapping IO and computation is limited to the first orientation of the first scattering vector. Second, the separation between data staging and scattering calculation provides each phase with *exclusive* access to MPI. Third, separating IO and computation allows more memory to be used for performance optimization in each phase. Finally, the maximum speedup of overlapping IO and computation here is inherently limited to a factor of 2. Fig. 1 illustrates the clear separation of the computational flow into an IO and computing phase.

3.1. Calculation schemes

Two major calculation schemes are supported: *All* scattering computes total scattering amplitudes, $A(\vec{q},t)$, and requires the trajectory data to be aligned by frames. *Self* scattering computes atomic scattering amplitudes, $a_n(\vec{q},t)$, and requires the trajectory data to be aligned by atoms. The alignment is performed during the initial IO phase and requires no changes to the data layout of the MD trajectory files.

3.1.1. All scattering

In *all* scattering the data are partitioned by frames. Thus, the number of frames determines the maximum partition size. The calculation is implemented in four phases: the computation of $A(\vec{q},t)$, gathering results on a single node, the post-processing of the full time signal, and final storage into the results buffer. The four phases constitute the scattering kernel, which computes the $\vec{q}$-dependent scattering signal $A(\vec{q},t)$, $F(\vec{q},t)$, or $F(\vec{q},\tau)$ (dependent on the DSP settings).

Scattering calculations which require orientational averaging, and thus need to integrate the scattering signal for a set of different scattering vectors $\{\vec{q}_i\}$ with the same scattering vector length $|\vec{q}|$, iterate over the scattering kernel. This allows advantage to be taken of two further improvements. The result after the first phase, $A(\vec{q}_i,t)$, is stored into an intermediate buffer and the first phase is then iterated until either the buffer is full or the iteration equals the size of the partition. The data in the resulting buffer are then transposed through a global MPI AlltoAll call, which places the full signal $A(\vec{q}_i,t)$ on a different node for each scattering vector $\vec{q}_i$, allowing the DSP phase and the storage into the results buffer to be performed in parallel. The computational flow for *all* scattering is illustrated in Fig. 2.

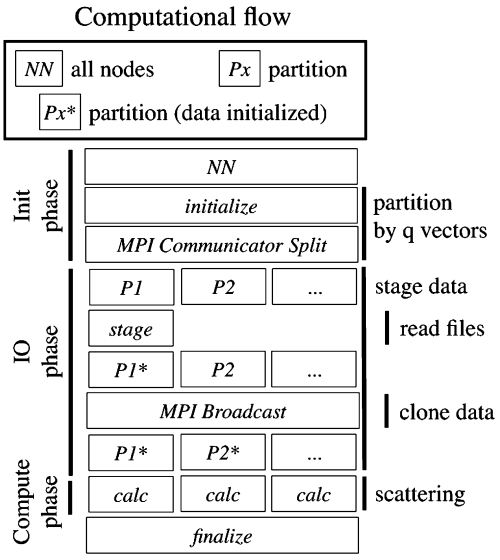


Fig. 1. Overall computational flow. The detailed instructions for the compute phase depend on the type of scattering and are illustrated in subsequent figures.

3.1.2. Self scattering

In *self* scattering the data are partitioned by atoms. Hence, the number of atoms determines the maximum partition size. In contrast to *all* scattering, *self* scattering does not require inter-node communication during the computation of the scattering signal. The scattering kernel thus contains only 3 phases, which are the computation of $a_n(\vec{q},t)$, post-processing of the full time signal, and a final store into the result buffer. Since the nodes operate independently on their assigned tasks, the only global communication is the final MPI reduce call for each independent $\vec{q}$ vector. This independence makes *self* scattering the ideal candidate for threading, which is discussed in Section 3.4. The computational flow for *self* scattering is illustrated in Fig. 3.

3.2. Data staging

In *all* scattering the required data layout coincides with the format in the trajectory file, which is optimal for parallel read. This way each node reads from a different section in the trajectory file and no data need to be exchanged between the nodes.

Table 2
Examples for the relation between the TYPE/DSP setting combination, the computed scattering function and the application.

TYPE	DSP	Resolution	Output	Function	Application
all	*square*	1	*fq*	$\langle F(\vec{q},t)\rangle_t$	Bragg diffraction
all	*square*	$\gtrsim 1000$	*fq*	$\langle\langle F(\vec{q},t)\rangle_t\rangle_\Omega$	Solution scattering / Powder diffraction
all	*autocorrelate*	any	*fqt*	$\langle\langle F(\vec{q},t,\tau)\rangle_t\rangle_\Omega$	Coherent dynamic scattering
all	*plain*	any	*fq2*	$\langle\lvert\langle A(\vec{q},t)\rangle_t\rvert^2\rangle_\Omega$	Elastic component of coh. dynamic scattering
self	*autocorrelate*	any	*fqt*	$\sum_n\langle\langle f_n(\vec{q},t,\tau)\rangle_t\rangle_\Omega$	Incoherent dynamic scattering
self	*plain*	any	*fq2*	$\sum_n\langle\lvert\langle a_n(\vec{q},t)\rangle_t\rvert^2\rangle_\Omega$	Elastic component of inc. dynamic scattering

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010

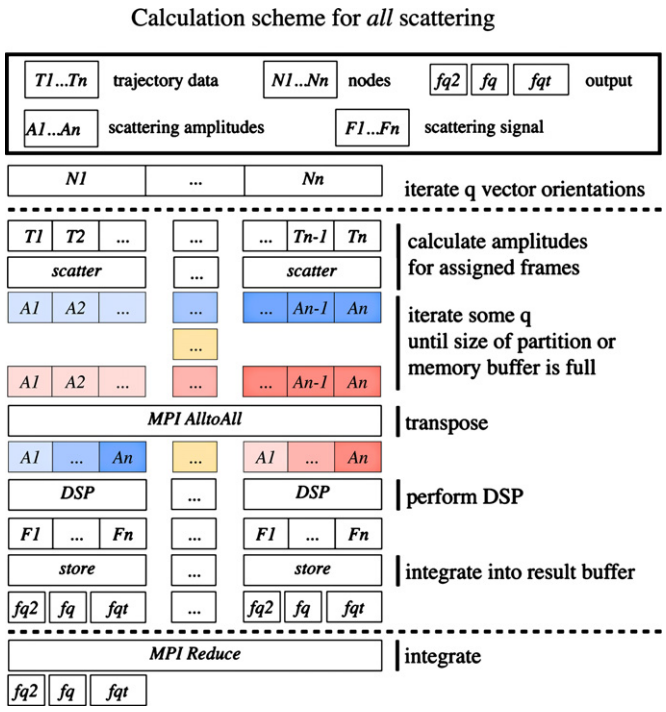


Fig. 2. Computational flow for *all* scattering. The computed scattering function is aggregated in the output fields *fqt*, *fq*, *fq2*.

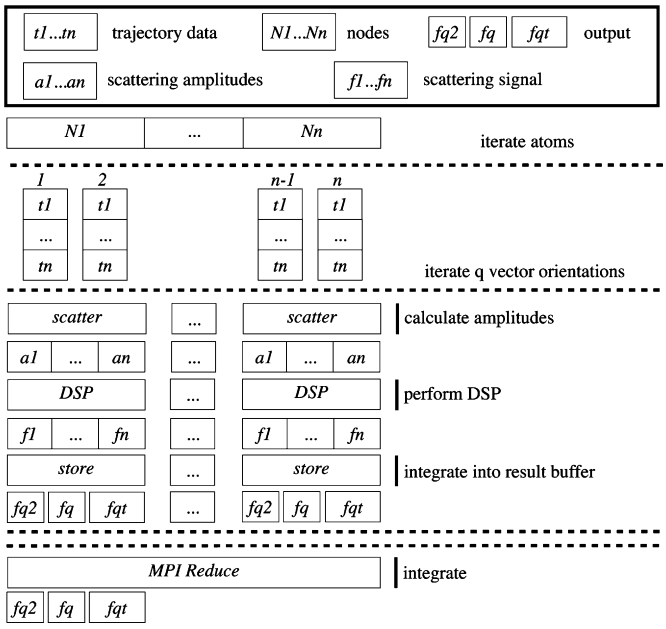
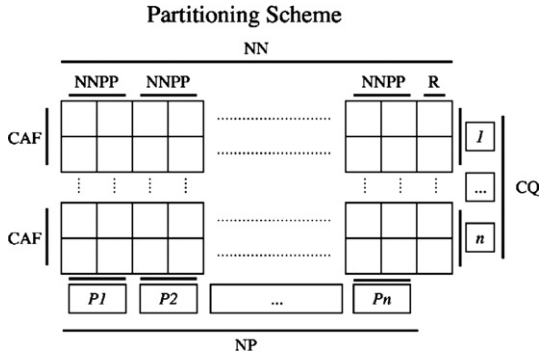


Fig. 3. Computational flow for *self* scattering. The computed scattering function is aggregated in the output fields *fqt*, *fq*, *fq2*.

In *self* scattering the data are decomposed by atoms, which means that each node has to read only a small portion of each frame. An embarrassingly parallel scheme would thus require rewriting the complete trajectory in a new format with the data aligned by atoms: otherwise, the IO bandwidth requirements would be increased by a factor equal to the partition size. Both options are unfavorable since rewriting the trajectory is a burden for usability and IO bandwidth is expensive. Thus, the data staging for *self* scattering is implemented in cycles of parallel read from the trajectory, in which each node reads a separate frame, followed by a global data transposition implemented as an MPI AlltoAll call. Since the communication through MPI is commonly carried out through a separate network with superior latency and bandwidth, the effective speed of staging the data for *self* scattering is comparable to that of *all* scattering (see performance results).



symbol	description
NN	all nodes
NP	number of partitions
$NNPP$	number of nodes per partition
NQ	number of independent q vectors
NAF	decomposition parameter (atoms or frames)
R	remaining nodes (not used)
CAF	cycles per q vector
CQ	q vector cycles

Fig. 4. Illustration of the underlying partitioning strategy (top) and definition of the symbols used (bottom). For a given partition size, *NNPP*, the utilization, U, of the available parallel bandwidth is pre-determined (see text).

3.3. Partitioning scheme

The calculation schemes for *all* and *self* scattering impose a limit on the number of cores that can be used in parallel. For *all* scattering this is equal to the number of frames, whereas for *self* scattering it is the number of atoms. However, most scattering calculations require the computation of the scattering signal for many independent scattering vectors $\vec{q}$. This allows an additional layer of parallelism to be added based on the number of independent scattering vectors. This results in a 2-dimensional partitioning scheme with the partition size being a parameter of choice with the following restrictions: Memory and Utilization.

Memory is a limiting factor since each partition has a full copy of the trajectory data. For trajectory data which exceed the available memory per node, the minimum allowed partition size is determined by the minimum number of nodes capable of holding the trajectory data in memory.

The utilization, U, is a performance characteristic of the chosen partition size and equals the fraction of time the cores are being used to compute scattering amplitudes. U can be precomputed:

$$U = \frac{NAF \cdot NQ}{NN \cdot CAF \cdot CQ} \tag{17}$$

$$CAF = \begin{cases} NAF \div NNPP & 0 \equiv NAF \bmod NNPP \\ (NAF \div NNPP) + 1 & \text{else} \end{cases}$$

$$CQ = \begin{cases} NQ \div NP & 0 \equiv NQ \bmod NP \\ (NQ \div NP) + 1 & \text{else} \end{cases} \tag{18}$$

The partitioning scheme and the origin of Eqs. (17) and (18) are illustrated in Fig. 4 together with the list of symbols.

Since the utilization can be computed in advance, the software calculates U for each possible decomposition and selects that partition size, *NNPP*, yielding the highest utilization. The default

threshold for the utilization is 95%, a number that may be set lower if required. Sometimes a particular partition size is favorable even though it may not yield the best utilization, U, as, for example, when the partition size should be equal to or a multiple of the number of cores per computer node. In cases where the partition size is equal to the number of cores per computer node, communication within a partition is local and avoids network latencies. These non-trivial aspects of the partition scheme make it hard to automatically select the overall best partition size for a given problem, which is why the user always has the option of selecting the partition size manually.

If the selected partitioning scheme results in more than one partition, the trajectory data are read by the first partition and subsequently cloned into the remaining partitions.

3.4. Threading

The parallel strategy outlined so far works well for most scattering calculations. However, the parallelization is inherently limited by the number of frames or atoms and by the number of independent scattering vectors. In the rare case where the user performs an *all* scattering calculation on a single frame and scattering vector with massive orientational averaging, the code does not parallelize through MPI. One option would be to implement another layer of MPI parallelization, in which the data are cloned throughout a set of nodes, each performing calculations on a few orientations. This strategy can be explicitly implemented by the user by integrating the signal from different runs of the software, each performed on a different random set of $\vec{q}$ vector orientations. However, in the light of recent developments in computer hardware threading is a better option to extent the scalability to cover massive orientational averaging. For this reason, the first phase in the scattering kernel for *all* scattering is implemented as a producer/consumer thread model. In this case it is recommended that the user sets the number of worker threads to a size similar to the number of cores per computer node, sets the partition size to 1 and allocates only one MPI process per compute node. Since each thread writes the signal into a separate buffer, threading is limited by the memory buffer available. If the partition size is larger than one, threading in *all* scattering is limited by the size of the partition due to data transposition requirements. Since in *self* scattering no global communication is necessary the whole scattering kernel can be threaded, thus allowing as many threads as memory buffer is available.

3.5. Algorithmic complexity

The scattering kernel has a complexity of $C = Q \cdot O \cdot N \cdot T$, where N is the number of atoms, T is the length of the trajectory, Q is the number of independent $\vec{q}$ vectors, and O is the number of orientations per vector. For each $\vec{q}$ vector orientation the computation of the autocorrelated time signal may be necessary (dependent on the scattering function). For the computation of the autocorrelated time signal, currently the FFTW implementation is used which scales with $T \cdot log(T)$. For *self* scattering this changes the complexity to $C_{self} = Q \cdot O \cdot N \cdot (T + T \cdot log(T))$, which makes the separation between the calculation of the scattering amplitudes and the autocorrelation explicit. For *all* scattering the complexity changes to $C_{all} = Q \cdot O \cdot (N \cdot T + T \cdot log(T))$. Parallelization through MPI allows partitioning the problem along Q and N for *self* scattering, and Q and T for *all* scattering. With NP being the number of partitions and $NNPP$ being the number of nodes per partition, the complexity is reduced to

$$C_{self} = \frac{Q}{NP} \cdot O \cdot \left(\frac{N}{NNPP} \cdot T + \frac{N}{NNPP} \cdot T \cdot log(T) \right) \tag{19}$$

$$C_{all} = \frac{Q}{NP} \cdot \left(O \cdot N \cdot \frac{T}{NNPP} + \frac{O}{NNPP} \cdot T \cdot log(T) \right) \tag{20}$$

The above equations indicate the scaling limits of the partitioning scheme, since the fractions Q/NP, $N/NNPP$, $T/NNPP$ and $O/NNPP$ cannot be less than 1. The automatic partitioning routine implemented in Sassena finds an optimal solution for the combination of $(NP, NNPP)$ for the particular scattering problem.

Additional (local) parallelization is possible by using threads. With NT being the number of threads per MPI process, the complexities change to

$$C^T_{self} = \frac{Q}{NP} \cdot \frac{O}{NT} \cdot \left(\frac{N}{NNPP} \cdot T + \frac{N}{NNPP} \cdot T \cdot log(T) \right) \tag{21}$$

$$C^T_{all} = \frac{Q}{NP} \cdot \left(\frac{O}{NT} \cdot N \cdot \frac{T}{NNPP} + \frac{O}{NNPP} \cdot T \cdot log(T) \right) \tag{22}$$

For *all* scattering NT is capped by $NNPP$ for $NNPP > 1$ and not restricted otherwise. The total number of available compute nodes is given by $NN = NP \cdot NNPP$. Both C_{self} and C_{all} are inversely proportional to NN, which implies perfect scalability as long as the problem size (Q, N, T) is large enough. In the limit of an infinitely large parallel machine the complexities converge to:

$$C^T_{self}(NP, NNPP, NT \to \infty) = T + T \cdot log(T) \tag{23}$$

$$C^T_{all}(NP, NNPP, NT \to \infty) = N + T \cdot log(T) \tag{24}$$

Thus, the complexity is reduced to a linear dependence on parameters of the simulated system (size and length). Further improvements in scalability are possible by schemes which employ alternative forms of local parallelism, *e.g.*, graphical processor units (GPUs), or by introducing an additional layer in the MPI partitioning scheme.

3.6. Implementation

The programming was performed with the C++ language and tested with the GNU C++ compiler v.4.4.2. Additional libraries include Boost C++ v1.43 [17], OpenMPI v1.4.3 [18], FFTW v.3.2.2 [19], xdrfile v1.1 [20], HDF5 v.1.8.5-patch1 [21] and libxml v.2.6.23 [22]. On CRAY supercomputers CRAY MPI was used instead of OpenMPI. The particular version numbers used for the libraries were those available at the time of writing and are likely to change in future versions. The software reads constants and mapping definitions from a set of XML files. Each run requires a configuration file written in XML. The scattering signal is written in HDF5 format.

4. Performance

Each time the software is executed it performs two main tasks: staging of the trajectory data and calculation of the scattering signal. The performance characteristics of these two tasks depend on different aspects of the underlying hardware, which is why they are discussed separately. Data staging is dependent on factors such as the number of available file servers, the file protocol, and the trajectory format, while scattering calculations depend mainly on the partitioning scheme, the available high performance network, and, in particular, on the type of scattering function. The performance was assessed by software internal timers. The single node computational efficiency of the scattering routine (which computes the scattering amplitudes) was measured with CrayPAT and was 850 MFlops at double precision with default optimization settings using the GNU C++ compiler, which corresponds to about 8.2% of the single node peak performance.

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010

Table 3
Systems used to run the performance benchmarks. Moldyn is a computational cluster owned by the Center for Molecular Biophysics at ORNL. Jaguar and Hopper are Cray supercomputers, administered by NCCS and NERSC, respectively.

Feature	System		
	Moldyn	Jaguar	Hopper
File System	NFS/Gigabit	LUSTRE	LUSTRE
Interconnect	Infiniband	Cray Seastar2+	Cray Gemini
Cores/Node	12	12	24
CPU Type	AMD Opteron 2431	AMD Opteron 2435	AMD Opteron 8378
GB RAM/Node	16	16	32
Nodes	100	18 688	6392

Table 4
Benchmark systems used characterized by their minimum job sizes. The symbols SL, MM and LS stand for small–large, medium–medium and large–small respectively. The benchmark files were tailored for minimum partition sizes of 8, 48 and 240.

System ID	Atoms	Benchmark 8			Benchmark 48			Benchmark 240		
		Frames	XTC	DCD	Frames	XTC	DCD	Frames	XTC	DCD
SL	613	400k	831M	2.8G	2M	4.1G	14G	5M	11G	35G
MM	120k	2k	930M	2.8G	15k	6.9G	21G	50k	24G	68G
LS	3.7M	60	845M	2.6G	500	6.9G	22G	2.5k	35G	107G

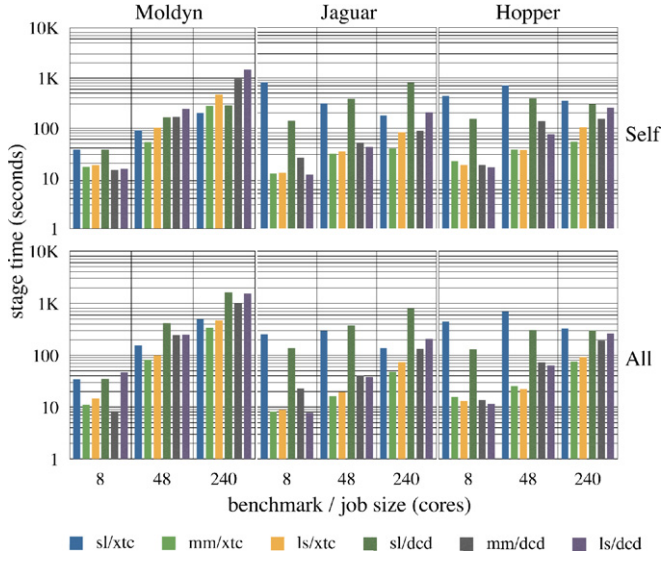


Fig. 5. IO performance results measured by time to completely load the trajectory data into memory for *all* and *self* scattering on Moldyn, Jaguar and Hopper for the three benchmark sets. The ordering of the data within the graphs corresponds to the order in the legend.

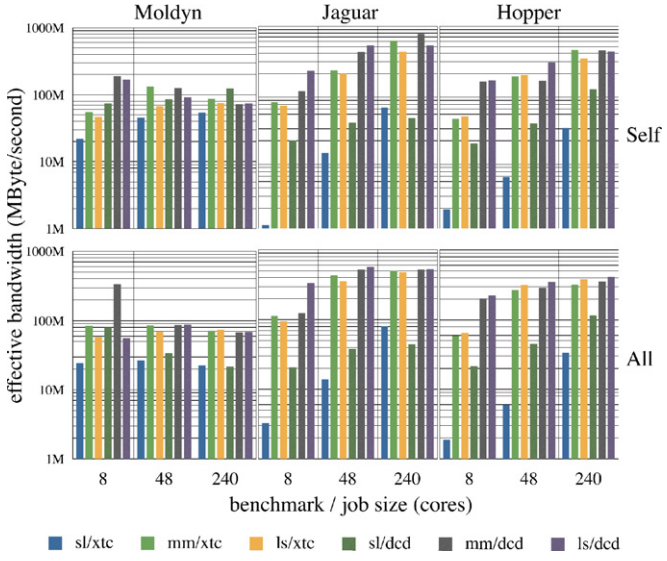


Fig. 6. IO performance results measured by the effective bandwidth achieved for *all* and *self* scattering on Moldyn, Jaguar and Hopper for the three benchmark sets. The ordering of the data within the graphs corresponds to the order in the legend.

4.1. IO

The IO performance is given by the time required to stage the trajectory data and was characterized by running a set of benchmarks on different computer systems, listed in Table 3. Moldyn is a computer cluster of 100 12-core compute nodes with an Infiniband network to perform MPI communication and possesses 2 file servers, one located on the head node and the other on a dedicated node, on which the benchmark data was placed. The file servers are connected to the compute nodes via Gigabit Ethernet. Jaguar is a Cray XT5 supercomputer at Oak Ridge National Laboratory and possesses 18 688 12-core compute nodes, a LUSTRE file system and a CRAY SeaStar2+ network for high performance communication. Hopper is a Cray XE6 supercomputer hosted by NERSC and possesses 6392 24-core compute nodes, a LUSTRE file system and a CRAY Gemini network. Jaguar and Hopper are similar architectures common for supercomputers, while Moldyn is the intermediate cost solution of a high fidelity computational cluster.

Three benchmark sets were prepared, corresponding to partition sizes of 8, 48 and 240 cores, which also determines the number of instances used to read the trajectory in parallel. Each benchmark set contains three molecular systems with distinct characteristics (frames/atoms) and file formats (XTC/DCD). The benchmark sets are listed in Table 4.

The IO performance results are given in Fig. 5. The performance was assessed as the time to finish the data staging phase and was determined as the minimum out of 10 trial runs for each case. The stage time was additionally renormalized by the file sizes to yield effective bandwidths in Mbyte/s, shown in Fig. 6.

Moldyn shows an approximately linear increase in data staging time with the partition size, stemming from the availability of only one file server and the bandwidth limitations of the Gigabit network. The time to stage data for *all* scattering is similar to that for *self* scattering. Jaguar shows significantly better scaling characteristics since more file servers and a better network are available. This causes the maximum effective bandwidth to approach the 1 Gbyte/s mark. However, for the SL benchmark files Moldyn actually performs better than Jaguar, especially for small

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010

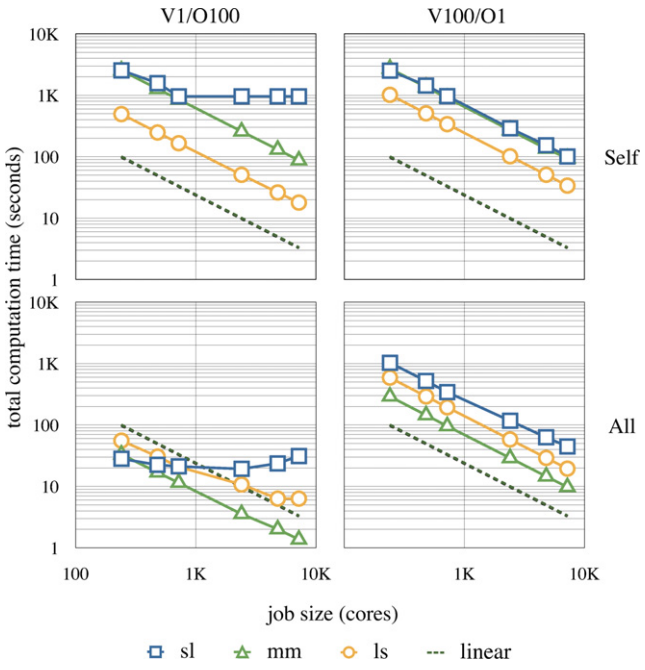


Fig. 7. Scaling results for Jaguar. The jobs sizes were 240, 480, 720, 2400, 4800 and 7200 cores. The partition size was chosen automatically to yield the best utilization. The scaling results for Moldyn and Hopper have similar characteristics (data not shown).

partition sizes. The SL system features a large number of small frames, thus generating much more file access requests than the other systems, and this seems to be a problem for the LUSTRE file system in the current case. Hopper exhibits IO characteristics similar to Jaguar, as expected since both supercomputers have very similar IO systems.

4.2. Scattering calculations

The computational demand depends on the type of scattering calculation to be performed. *All* scattering requires global MPI communication during the calculation because the data are decomposed by frames and the DSP step requires the full time signal. Thus, for each orientation of the q vector, the signal must be aggregated on one node, which then performs post-processing operations and puts the result into local buffers. *Self* scattering does not require data exchange during the calculation since the trajectory data has been placed accordingly. However, *self* scattering requires a post-processing operation for each single atom, *i.e.*, the computation of an autocorrelation, which can become significant for long trajectories (FFTW scales with $O(T \cdot log(T))$).

The performance characteristics were sampled with a set of 3 benchmark files and 4 variations of the scattering function, consisting of *all* vs. *self* scattering and a calculation for one scattering vector with 100 orientations (V1/O100) and 100 scattering vectors with one orientation each (V100/O1). The DSP setting was set to *autocorrelate*, which computes dynamic scattering functions and allowed the investigation of the impact of significantly different computational costs for DSP (complexity $O(T \cdot log(T))$) on the scalability for the three systems SL, MM and LS. Fig. 7 shows the scaling results for Jaguar. Hopper and Moldyn have similar scaling characteristics (data not shown).

The heterogeneity in the type of function to compute makes it difficult to derive a single performance measure. However, from Fig. 7, a few observations are apparent. Computing the signal for the SL system in case of V1/O100 does not scale well. For *self* scattering the reason is trivial since the partitioning is limited by the

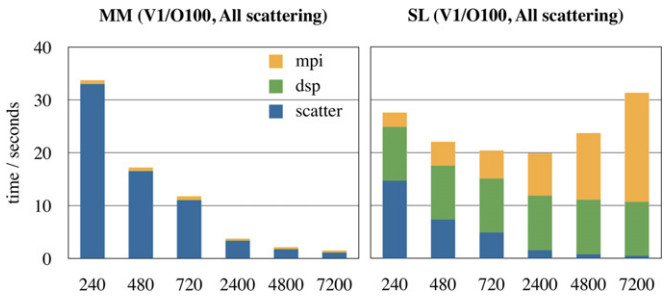


Fig. 8. Scattering calculation time for the SL and MM systems in the V1/O100 case decomposed into individual components of the scattering kernel. The time spent in the DSP phase and MPI communication is negligible for the MM and significant for the SL system.

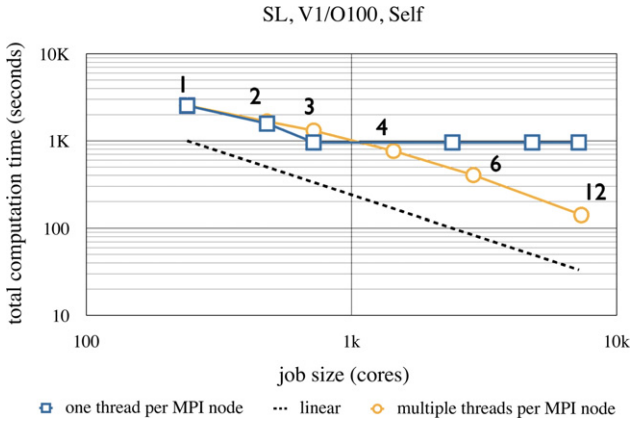


Fig. 9. Scalability result using multiple threads for V1/O100 *self* scattering of the SL system. For clarity the result without using multiple threads is also shown.

number of atoms. For *all* scattering the reason is more complex and has been investigated by decomposing the total calculation time into its components, shown in Fig. 8.

The decomposition shows that the time to compute an autocorrelation becomes significantly larger than the time to compute the scattering amplitudes for larger partition sizes, and therefore there is no effective speed-up when increasing the number of cores. Instead the increase in the number of cores increases the MPI overhead in the global communication until the overhead becomes dominant.

The scalability of V1/O100 *self* scattering for the SL system can be increased by employing threads, which is illustrated in Fig. 9. By placing a single MPI process on each 12-core compute node, the partition can span $12 \cdot 631$ cores instead of 631. The cores on each compute node are then utilized using threads.

5. Validation

The code was validated against output from existing software (SASSIM [4], nMoldyn [9]). Sassena also supports to combine the molecular dynamics coordinates with either deterministic or stochastic motions, allowing to study the interplay between realistic and idealized motions, and this facility was used to validate the overall agreement with various theoretical predictions. For example, oscillatory motions of a particle yield oscillations in the associated intermediate scattering function $F(\vec{q}, t)$ with the correct frequencies, and Brownian motion leads to a single exponential decay in $F(\vec{q}, t)$ with a decay fitting constant leading to the correct diffusion constant, *i.e.*, that used to generate the Brownian motion in the first place. The validation results are provided online.[2]

[2] http://www.sassena.org.

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010

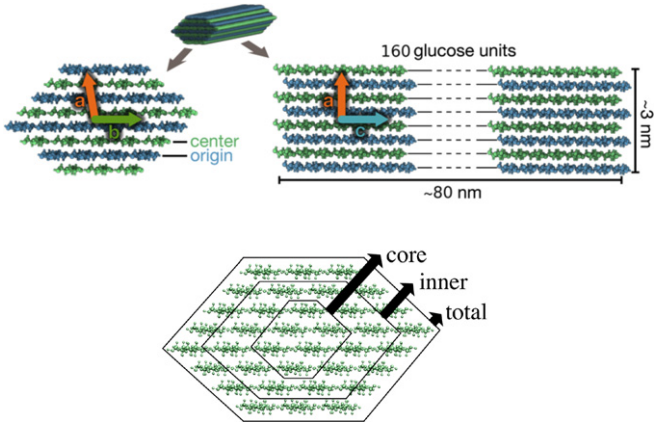


Fig. 10. Cellulose fiber I_β structure with 36 chains and 160 glucose monomers per chain, totaling 121 068 atoms.

6. Application

The utility of the software is illustrated by two examples. The target data are the MD simulation of an I_β crystalline cellulose elementary fibril with the initial starting configuration shown in Fig. 10. Cellulose is a biopolymer formed mainly in plant cell walls [23] that is of high potential for biofuel production. The atomic crystal coordinates were taken from Ref. [24] and the fiber model is from Ref. [25]. The model was simulated using GROMACS [26] and the CHARMM force field for cellulose [27] at constant temperature (300 K) and pressure (1 atm) with simulation parameters detailed in Ref. [11]. The model was simulated for 20 ns, with a subsequent simulation also performed for 200 ps at a recorded trajectory time step of 1 fs (200 K frames).

6.1. Wide angle X-ray scattering (WAXS)

WAXS is a scattering technique which probes structural correlations on the atomic scale. The time-resolved characteristic function, $F(q,t)$, for orientationally-averaged WAXS is given by

$$F(q,t) = \left\langle F(\vec{q},t)\right\rangle_{\Omega} \tag{25}$$

The most prominent WAXS scattering peak for cellulose is the (200) peak which appears at $|\vec{q}| = 1.6/$ Å, and marks the sheet-to-sheet distance along unit cell base vector $\vec{a}$ in crystalline cellulose [28] (see Fig. 10).

The WAXS diagram (120 points) for cellulose was calculated from the simulation model for the initial starting configuration and the final configuration after 20 ns of MD simulation time and is shown in Fig. 11. During the simulation there is a marked shift of the primary peak from 1.6 Å^{-1} to less than 1.45 Å^{-1} consistent with an increase of the $\vec{a}$ unit cell base vector of the cellulose. The fact that the peak shift occurs for the core and inner chains in the fiber cross-section indicates that the structural change happens throughout the whole fiber.

The convergence of the WAXS intensities was probed by varying the number of $\vec{q}$ vector orientations between 50 and $50 \cdot 10^4$ and is shown in Fig. 12. For each resolution (number of $\vec{q}$ vector orientations) the scattering diagram was calculated 10 times. The error E is given as the average pairwise difference between the estimated WAXS intensities:

$$E = \left\langle \sum_{k=1}^{120} \left| \left\langle F(q_k)\right\rangle_{\Omega_i} - \left\langle F(q_k)\right\rangle_{\Omega_j} \right| \right\rangle_{ij} \tag{26}$$

The convergence analysis of the WAXS pattern for crystalline cellulose shows that a large number of $\vec{q}$ vector orientations

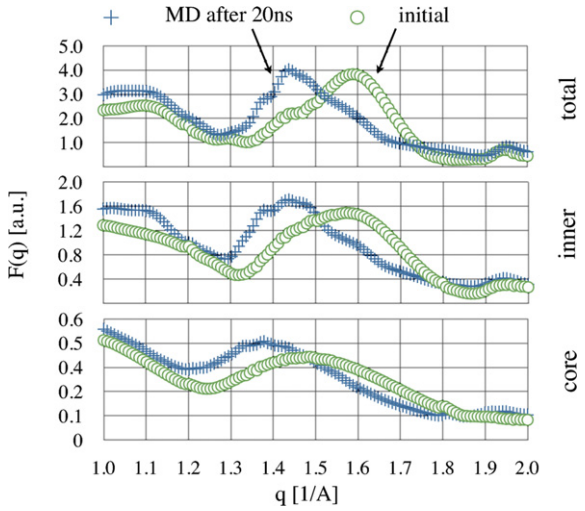


Fig. 11. WAXS diagram of cellulose before and after MD simulation for different cross-sectional fractions of the cellulose fiber.

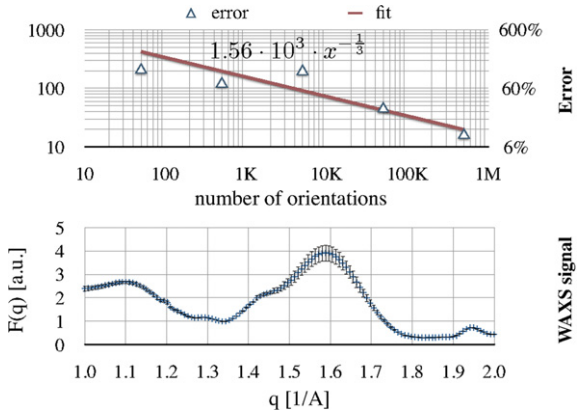


Fig. 12. (Top) Error of convergence and (bottom) average X-ray wide angle scattering diagram for crystalline I_β cellulose, averaged for 10 calculations for different random sets of $5 \cdot 10^5$ $\vec{q}$ vector orientations.

(> 100 K) is necessary to achieve an average accuracy of about 10% (the error varies with $|\vec{q}|$). This requirement is due to the crystalline nature of the sample, which causes the scattering intensities to become focused in $\vec{q}$ space. Therefore, a larger number of orientations are needed to accurately approximate the correct value of the spherical integral. For systems which exhibit less symmetry the required number of orientations is usually considerably less.

The computations were performed on the Jaguar Cray XT5 Petaflop Supercomputer, with a partition size of 120, allocating one MPI node to each core, with the exception for the calculation at a resolution of $5 \cdot 10^5$ which was performed with a job size of 1440, placing each MPI node on a separate compute node and using 12 threads per node. The performance P, given as the number of vectors and frames processed per time unit and core, was $P = 70.6$ frame · vector/s/core.

6.2. Inelastic X-ray scattering (IXS)

IXS is a scattering technique which resolves X-ray photon energies after the scattering event and allows the vibrational density of states of the sample to be deduced. In crystals the $\vec{q}$ vector dependence of the transferred energy of the X-ray photons are quantized solutions to the vibrational spectrum of the lattice and follow an anisotropic dispersion relation. The IXS signal, $S(q,E)$, can be calculated as the Fourier transform of the coherent intermediate scattering function and a subsequent renormalization with

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010

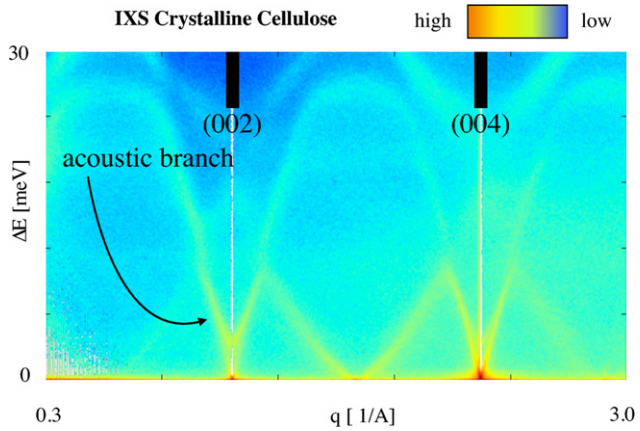


Fig. 13. Calculated IXS signal from a 200 ps MD simulation of the I_β crystalline cellulose model. The recorded time step was 1 fs, requiring the analysis of 200 K frames.

the energy–frequency relationship, $E = hv$, where h is Plank's constant. The characteristic function is:

$$S(q, v) = FT\left\{\left\langle\left\langle F(\vec{q}, t, \tau)\right\rangle_t\right\rangle_\Omega\right\} \tag{27}$$

IXS experiments on aligned cellulose fiber samples were used to deduce the velocity of sound along the fiber axis [29]. However, the experimental IXS data are very noisy and preclude the identification of different phonon branches. The calculated IXS signal for the simulated cellulose reveals a characteristic phonon dispersion pattern with a clear acoustic branch originating from the center of the (002) Brillouin zone, indicated in Fig. 13. The detailed physics of this calculated IXS is currently under investigation.

The IXS intensities were calculated on Jaguar using 960 cores, allocating one MPI process per core. The spectrum contains 1000 independent q vectors and is based on 200 K frames. The performance was $P = 74.4$ frame · vector/s/core.

7. Future development

The analysis of the scalability and the performance of Sassena shows strengths and weaknesses of the current implementation. While the scalability results are satisfying, the IO can be a major bottleneck in calculations on large datasets. On platforms such as the Jaguar and Hopper supercomputers, the IO system is able to scale well with the problem size, effectively reaching $\frac{1}{2}$ Gbyte/s. However, for trajectories with millions of frames the reading of the data is organized into millions of separate IO requests, which creates a bottleneck even for these platforms. The single node performance of the scattering kernel is a major candidate for subsequent improvements. The following is an outline of future developments for improving the overall performance of the software.

The IO performance results illustrate that supercomputing platforms feature superior IO bandwidth but are limited with respect to IO requests, which is why they underperform for the SL system. On platforms which are IO bandwidth limited all three benchmark systems show similar performance. Since the MD data files are organized by frames, the reading of neighboring frames may be bundled into the same IO request. This may be facilitated by introducing a preloading procedure, which reads the file data into an intermediate memory buffer from which the frames are extracted. These changes are feasible and can be implemented for both scattering calculation types.

Multiple options exist to increase the computational performance. Use of special instruction sets such as SSE or the use of GPUs are direct ways to yield better performance. Special purpose methods for improving the convergence of orientational averaging may also be considered, in particular adaptive integration algorithms such as Markov Chain Monte Carlo [30] or Multipole Expansion [31].

The data staging and calculation schemes may be used as templates for other kinds of analysis not directly involving scattering, such as the computation of mean-square displacements. Since the software is available as open source the code may either serve as an inspiration for the parallelization of other software packages or may be extended to support required alternative types of calculation.

8. Availability

The software is available online as open source under the GPL license model (http://www.sassena.org). Additional documents, *e.g.*, configuration file manuals and tutorials, are also available online.

Acknowledgement

This project was funded by Grant Number MCB-0842871 from the National Science Foundation. This research used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725, and used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Benjamin Lindner thanks Liang Hong and Roland Schulz for inspirational discussions about scattering theory and computational methods, respectively. Also thanks to Franci Merzel who helped Benjamin Lindner getting started with scattering calculations.

References

[1] O. Glatter, The inverse scattering problem in small-angle scattering, in: Neutrons, X-rays and Light. Scattering Methods Applied to Soft Condensed Matter, Elsevier Science, 2002, pp. 73–103.

[2] Frank Noé, Sören Doose, Isabella Daidone, Marc Löllmann, Markus Sauer, John D. Chodera, Jeremy C. Smith, Dynamical fingerprints for probing individual relaxation processes in biomolecular dynamics with simulations and kinetic experiments, PNAS 108 (12) (March 2011) 4822–4827.

[3] Jeremy C. Smith, G.R. Kneller, Combination of neutron scattering and molecular dynamics to determine internal motions in biomolecules, Molecular Simulation 10 (2–6) (1993) 363–375.

[4] Franci Merzel, Jeremy C. Smith, SASSIM: a method for calculating small-angle X-ray and neutron scattering and the associated molecular envelope from explicit-atom models of solvated proteins, Acta Crystallographica Section D Biological Crystallography 58 (2002) 242–249.

[5] F. Merzel, Jeremy C. Smith, Is the first hydration shell of lysozyme of higher density than bulk water? Proceedings of the National Academy of Sciences 99 (8) (April 2002) 5378–5383.

[6] P. Faure, A. Micu, D. Perahia, J. Doucet, Jeremy C. Smith, J.P. Benoit, Correlated intramolecular motions and diffuse X-ray scattering in lysozyme, Nature Structural & Molecular Biology 1 (2) (February 1994) 124–128.

[7] A. Micu, Jeremy C. Smith, SERENA: a program for calculating X-ray diffuse scattering intensities from molecular dynamics trajectories, Computer Physics Communications 91 (1–3) (September 1995) 331–338.

[8] Lars Meinhold, Jeremy C. Smith, Fluctuations and correlations in crystalline protein dynamics: a simulation analysis of staphylococcal nuclease, Biophysical Journal 88 (April 2005) 2554–2563.

[9] T. Róg, K. Murzyn, K. Hinsen, G.R. Kneller, nMoldyn: a program package for a neutron scattering oriented analysis of molecular dynamics simulations, Journal of Computational Chemistry 24 (5) (April 2003) 657–667.

[10] D. Svergun, C. Barberato, CRYSOL—a program to evaluate X-ray solution scattering of biological macromolecules from atomic coordinates, Journal of Applied Crystallography 28 (1995) 768–773.

[11] Roland Schulz, Benjamin Lindner, Loukas Petridis, Jeremy C. Smith, Scaling of multimillion-atom biological molecular dynamics simulation on a petascale supercomputer, Journal of Chemical Theory and Computation 5 (June 2009) 2798–2808.
[12] Marc Bee, Quasielastic Neutron Scattering: Principles and Applications in Solid State Chemistry, Biology and Material Science, Adam Hilger, Dec. 1988.
[13] Th. Zemb, P. Lindner, Neutrons, X-rays and Light: Scattering Methods Applied to Soft Condensed Matter, Elsevier, Boston, Amsterdam, 2002.
[14] G.L. Squires, Introduction to the Theory of Thermal Neutron Scattering, Dover Publications, New York, 1996, 1978.
[15] J.M. Cowley, B.L.M. Peng, I.G. Ren, J.S.L. Dudarevc, M.J. Whelanc, Parameterizations of electron atomic scattering factors, International Tables for Crystallography, C (Ch. 4.3), 2006, p. 262.
[16] Varley F. Sears, Neutron scattering lengths and cross sections, Neutron News 3 (1992) 26–37.
[17] B.O.O.S.T. Developers, Boost C++ Libraries, March 2011, http://www.boost.org.
[18] Open MPI Developers, Open MPI, March 2011, http://www.open-mpi.org.
[19] F.F.T.W. Developers, FFTW v3.2.2 Library, March 2011, http://www.fftw.org/.
[20] G.R.O.M.A.C.S. Developers, Gromacs XDR File v1.1, March 2011, http://www.gromacs.org.
[21] The HDF5 Group, HDF5, March 2011, http://www.hdfgroup.org/HDF5/.
[22] LibXML developers, libxml, March 2011, http://xmlsoft.org/.
[23] Daniel J. Cosgrove, Growth of the plant cell wall, Nature Reviews Molecular Cell Biology 6 (11) (November 2005) 850–861.
[24] Y. Nishiyama, P. Langan, H. Chanzy, Crystal structure and hydrogen-bonding system in cellulose 1 beta from synchrotron X-ray and neutron fiber diffraction, Journal of the American Chemical Society 124 (31) (2002) 9074–9082.
[25] She-You Ding, Michael E. Himmel, The maize primary cell wall microfibril: a new model derived from direct visualization, Journal of Agricultural and Food Chemistry (2006).
[26] Berk Hess, Carsten Kutzner, David van der Spoel, Erik Lindahl, GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation.
[27] Michelle Kuttel, John W. Brady, Kevin J. Naidoo, Carbohydrate solution simulations: producing a force field with experimentally consistent primary alcohol rotational frequencies and populations, Journal of Computational Chemistry 23 (13) (2002) 1236–1243.
[28] Samira Elazzouzi-Hafraoui, Yoshiharu Nishiyama, Jean-Luc Putaux, Laurent Heux, Fredric Dubreuil, Cyrille Rochas, The shape and size distribution of crystalline nanoparticles prepared by acid hydrolysis of native cellulose, Biomacromolecules 9 (1) (2008) 57–65.
[29] Imke Diddens, Bridget Murphy, Michael Krisch, Martin Müller, Anisotropic elastic properties of cellulose measured using inelastic X-ray scattering, Macromolecules 41 (November 2008) 9755–9759.
[30] Siddhartha Chib, Siddhartha Chib, Edward Greenberg, Understanding the Metropolis–Hastings algorithm, The American Statistician 49 (4) (November 1995) 327–335.
[31] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, Dover, New York, Mar. 1970.

Please cite this article in press as: B. Lindner, J.C. Smith, Sassena — X-ray and neutron scattering calculated from molecular dynamics trajectories using massively parallel computers, Computer Physics Communications (2012), doi:10.1016/j.cpc.2012.02.010