マルチノードGPUを利用した
大規模な高分子系のMD計算の事例紹介

hp210134 全原子MD計算によるポリフッ化ビニリデンと
ポリエチレンの結晶化挙動の解明
hp210133 水中Tetra-PEGゲルの負のエネルギー弾性の分子論的解明
jh210035 GPUの高速並列計算で実現する交差禁止制御可能な
高分子シミュレータの開発

防衛大　萩田

2021.8.30月曜日 1655-1715

第2回スーパーコンピュータ「不老」ユーザ会

1 高分子物理・材料でのGPU活用

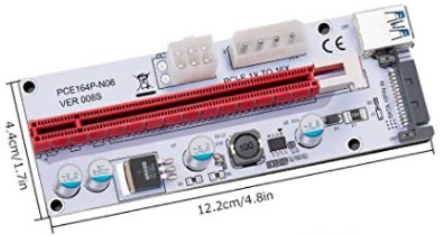
- この1年程度で、GPUを活用できる研究環境が、すごく整った。 → エコシステムの完成？
  - ハード
    - V100やA100の、高性能なマルチGPU環境
    - RTX 3090などの、高性能なゲームGPUの普及
  - ソフト　＜オープンソースのソフトウェア＞
    - “Single”-GPUの性能向上（単精度＋GPU内計算）
    - マルチGPUでも性能が出るようになった。

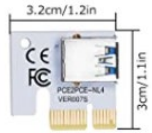
	Single GPU性能	マルチ GPU	ばね ビーズ	DPD	UAMD/ 全原子	ReaxFF
Gromacs	◎	◎	ー	ー	◎	ー
HOOMD-blue	◎	△？	◎	◎	△	ー
LAMMPS	△	○	○	○	△	◎

2

Single-GPUの活用

- ホスト-デバイス間通信の少ないSingle-GPU利用では、マイニング的なハードが使える。
  - PCにGPU1枚追加。
    - PCI-e 1xのライザーカード
    - 電源＋スイッチ





ソフト面のおまけ情報

- OpenACCも、入門者には、かなり使える。
  - 実務を改善する程度の“相対的”高速化を狙う場合。
  - GTX1070でも、いい感じである。（驚き）
    - 逆に、A100やV100では、勿体ない感じ。

- 「3DFFT＋多重ループで足し込み」が繰り返される解析用ユーザーコードで、特に有効だった。
  - 東大 GPUキャンプの成果。

K21-08-158-05 チーム

- 目標設定と、達成状況

目標設定	達成状況／見通し
2次元散乱像（SPring-8）から、ナノ粒子の3D構造変化を推定するPM-2DpRMC法のコードを、A100x8ノードで最適化。	Single-GPUでは満足 Multi-GPU FFTが未達
現CPUコードよりも、大規模な系を高速に扱えるようにする。	性能向上で、達成可能
GPU対応もしたコードを、論文化で一般公開する。	Full-3DFFTで公開可能

- 実施内容 ① cuFFT+OpenACCの、ミニコード(C++)で作業。
  ② プロダクトコードでの、最適化
  - OpenACC+cuFFTの作業と並行して、Intel CPUコードの最適化も実現。

	べた書き最適化 sparse-3D-FFT		未最適化 full-3D-FFT	cuFFT OpenACC	（左） 単精度化	sparse-3D-FFT OpenACC暫定
秒 500 trials	SR1600 32smp	CPU 72omp	CPU 72omp	A100	A100	A100
MainLoop	1.5くらい	0.567	**3.693**	1.569	**0.886**	0.654
3dFFT部			0.977	1.399	0.758	
畳込計算部		0.424	3.693	0.147	0.107	0.574

K21-08-158-05 チーム

- cufft+OpenACC
  - ループの開始前

```
!$acc data copyin(pos) copyin(sq1exp) copy(rmcImg) copy(reImg) create(co2Img) copyout(sq1)
create(sq2d) create(nsq2d) create(psq2d) create(pnsq2d) create(move)
```

  - ループ内の3d-FFT

```
ierr = cufftPlan3D(iplan1,ngrid0,ngrid1,ngrid1,CUFFT_D2Z)
        ierr = ierr + cufftSetStream(iplan1,acc_get_cuda_stream(acc_async_sync))
        !$acc update device(reImg)
        !$acc host_data use_device(reImg,co2Img)
        ierr = ierr + cufftExecD2Z(iplan1,reImg,co2Img)
        !$acc end host_data
        ierr = ierr + cufftDestroy(iplan1)
```

K21-08-158-05 チーム

– 畳み込み計算部分

```
!$acc kernels
sq2d(:,:)=0.0d0
nsq2d(:,:)=0.0d0
!$acc end kernels

!$acc kernels
psq2d(:,:,0:ngrid1/2)=0.0d0
pnsq2d(:,:,0:ngrid1/2)=0.0d0
!$acc end kernels
```

OpenMPの
first-touch的な処理

```
!$acc kernels
!$acc loop independent gang
do iz=0,ngrid1/2
  dqz=dqya(iz)
  if(iz.eq.0.or.iz.eq.ngrid1/2) then
    !$acc loop seq
    do iy=0,ngrid1-1
      dqy=dqya(iy)
      i2=int((sqrt(dqy+dqz)+0.5d0)*dqfft1/dq)
      if(i2.lt.iqmax1-2) then
        !$acc loop independent vector
        do ix=0,ixm
          dqx=dqxa(ix)
```

```
!$acc atomic
pnsq2d(i1,i2,iz)=pnsq2d(i1,i2,iz)+dconst(ix)*w00/wsum
!$acc atomic
pnsq2d(i1+1,i2,iz)=pnsq2d(i1+1,i2,iz)+dconst(ix)*w10/wsum
```

おまけ情報

- GPUキャンプでの成果

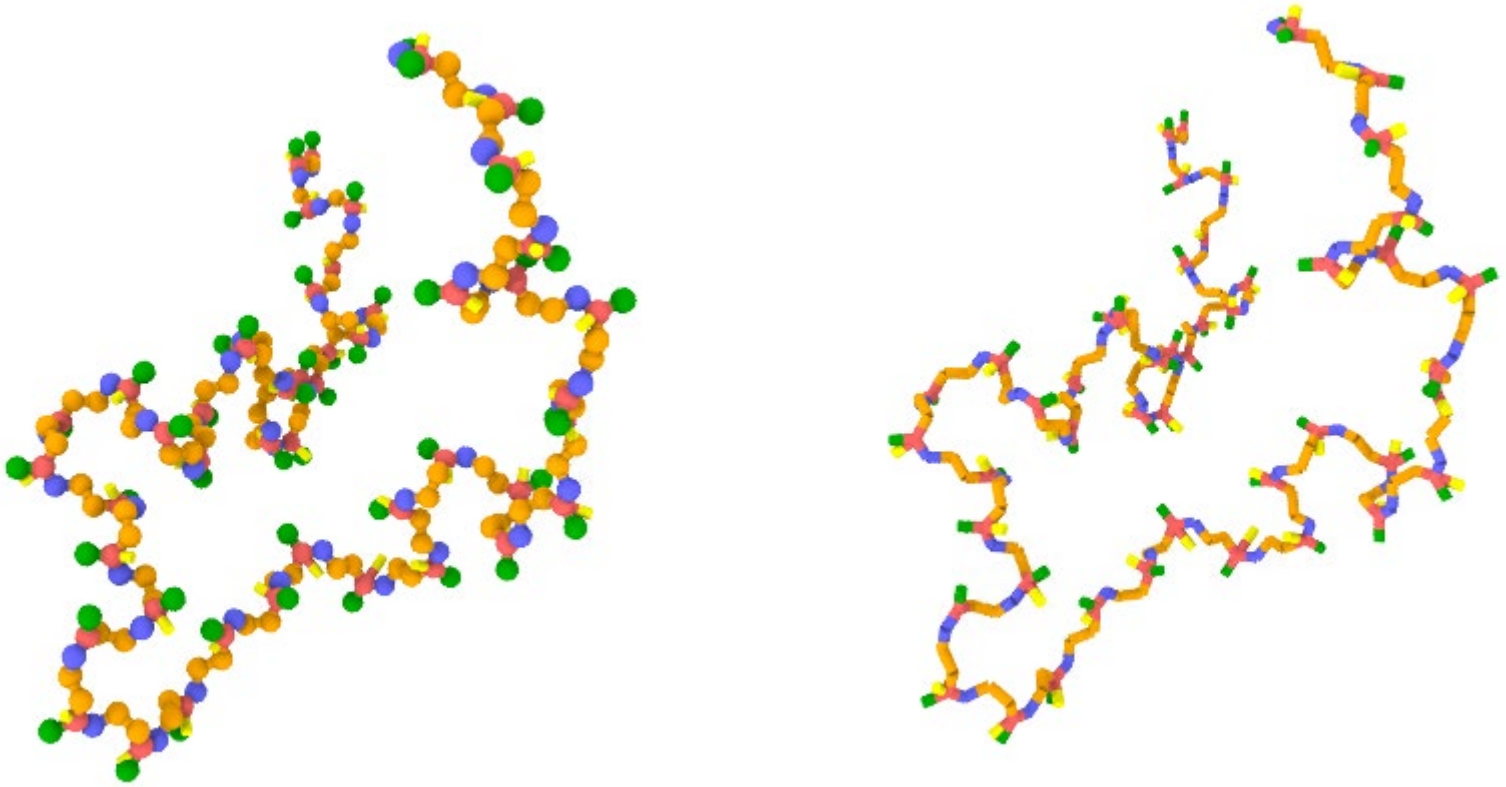
	べた書き最適化 sparse-3D-FFT		未最適化 full-3D-FFT	cuFFT OpenACC	（左） 単精度化	sparse-3D-FFT OpenACC暫定
秒 500 trials	SR1600 32smp	CPU 72omp	CPU 72omp	A100	A100	A100
MainLoop	1.5くらい	0.567	**3.693**	1.569	**0.886**	0.654
3dFFT部			0.977	1.399	0.758	
畳込計算部		0.424	3.693	0.147	0.107	0.574

- その後のフォローアップ

	秒 500 trials	A100	RTX3090	RTX3070	RTX2070	GTX1070
cuFFT OpenACC	MainLoop	1.569	1.887	2.856	2.788	3.723
	3dFFT部	1.399	1.383	2.364	2.469	3.148
	畳込計算部	0.147	0.494	0.480	0.308	0.558
（上） 単精度化	MainLoop	**0.886**	0.938	1.585	1.312	1.931
	3dFFT部	0.758	0.746	1.295	0.983	1.409
	畳込計算部	0.107	0.183	0.278	0.317	0.504

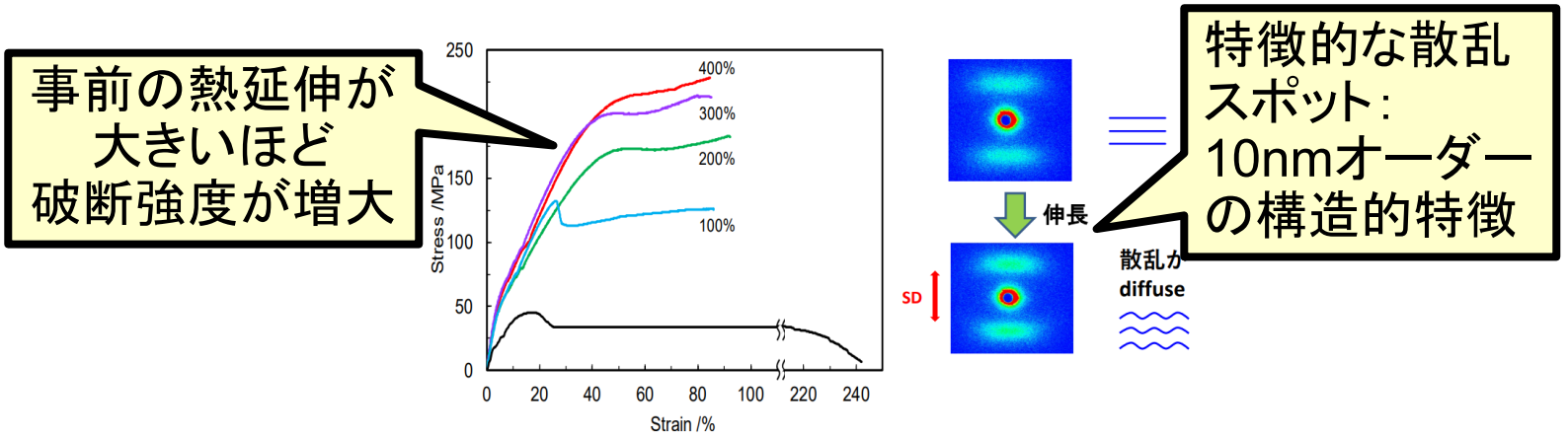
さて本題

- 高分子材料　・・・　構造と力学特性の関連を、MD計算で予測したい。
  - 実験とデータ同化し、実験の系を制御したい。



背景

- 結晶性高分子 （ポリフッ化ビニリデン）

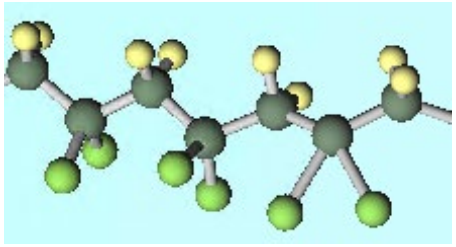


- ミルフィーユ構造を持つ材料は、熱延伸で強化
  - 金属／セラミック／高分子で共通： 強化原理は？
  - 高分子の分子レベルでの強化メカニズムは未解明
- 科研費・**新学術**領域 （東大・工 阿部先生代表）
  - 公募研究 萩田 「キンク強化が期待される結晶性高分子材料の分子論的解明」 → MD計算で解明

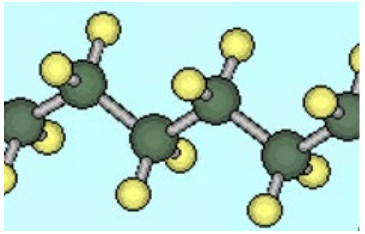
全原子MDと、ユナイテッドアトムMD

- ポリフッ化ビニリデンと、ポリエチレンで、物性差
  - 全原子MDで、差を見いだす。

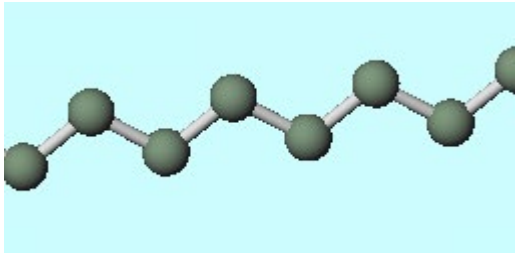
(a) PVDF
$(-CH_2-CF_2-)_n$



(b) PE
$(-CH_2-CH_2-)_n$

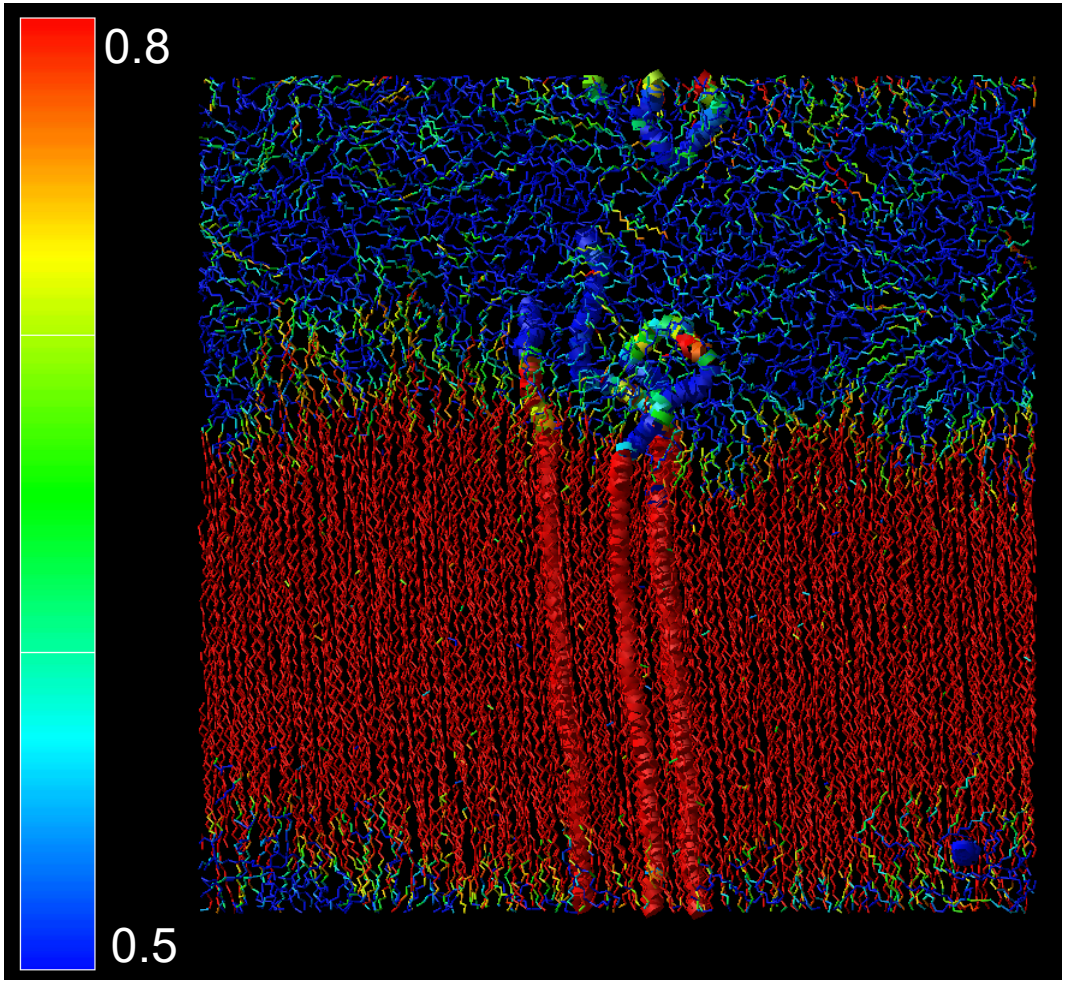


- 上記の前段階研究として、ミルフィーユ構造での熱延伸強化の大枠を掴む。
  - ユナイテッドアトムMDで、粗視化高速計算。
    - 骨格以外の原子省略
    - クーロン力なし



ミルフィーユ構造の分子描像

- 結晶の部分は、配向が、$\cos\theta > 0.8$。
  - 事前延伸により配向させた系での結晶化

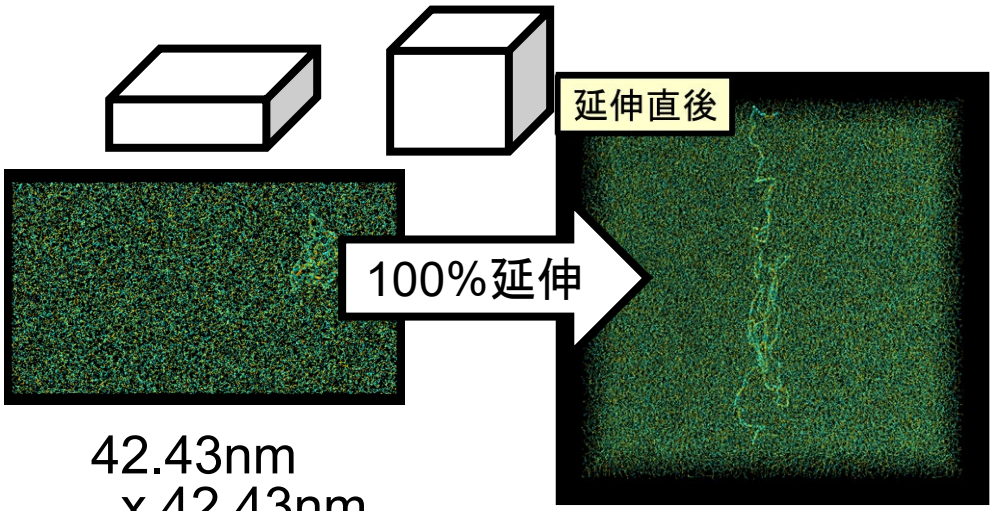


柔らかい
「アモルファス層」

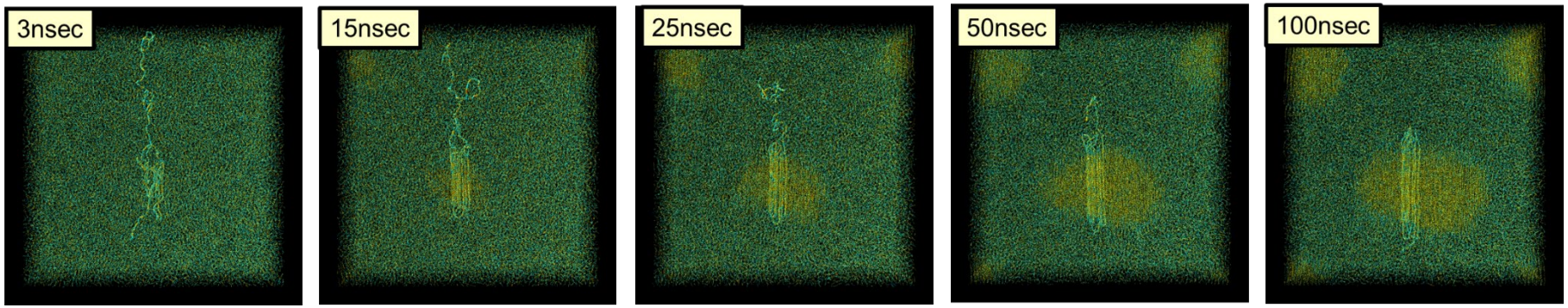
硬い
「結晶層（配向した鎖）」

熱延伸した結晶性高分子の計算

- 事前の延伸



- 静置して、結晶化を観察



一部修正

GromacsでのGPU計算@名大

- 100万原子のUAMD模型
  - 同じバイナリ、同じスクリプトで、４月の計測と、８月の計測で大きな差がある。（なぜ？）

	ノード数	MPI並列	#_OMP	ns/day (4月計測)	ns/day (8月計測)	GPU数
cx-single	1	4 (tMPI)	6	111.940	103.166	4
cx-single	1	8 (tMPI)	5	143.867	131.148	4
cx-single	1	4 (oMPI)	6	120.441	112.560	4
cx-small	4	16 (oMPI)	6	348.527	306.190	16
cx-small	8	32 (oMPI)	6	521.053	402.881	32
cxgfs-middle	16	64 (oMPI)	6	742.913	551.185	64
cx-large	32	128 (oMPI)	6	**1004.919**	X	128

tMPI: thread-MPI(ノード内)、oMPI: openmpi-4 cuda-aware

一部修正

GromacsでのGPU計算@A100

- 100万原子のUAMD模型

ノード数	Cuda-aware MPI並列	東大BDEC (ns/day)	阪大SQUID (ns/day)	GPU数
1	4(oMPI)	164.246	—	4
1	6(oMPI)	216.820	253.592	6
1	8(oMPI)	262.641	291.100	8
2	16(oMPI)	414.496	433.492	16
4	32(oMPI)	626.704	584.688	32
8	64(oMPI)	549.630	**680.177**	64

(8月)名大のV100は、
64 GPUで、
551.185ns/days
なので、問題は謎！
（電力設定？？）

(4月)名大のV100が、
64 GPUで、
742.913ns/days
128GPUで、
1004.919ns/days
で速かったですけど
問題は、未解決。

tMPI: thread-MPI(ノード内)

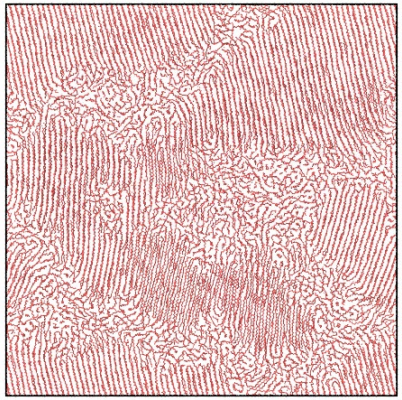
oMPI: openmpi-4 cuda-aware

ノード数	Thread-base MPI並列	東大BDEC (ns/day)	GPU数
1	4(tMPI)	92.306	4
1	6(tMPI)	165.748	6
1	8(tMPI)	152.069	8

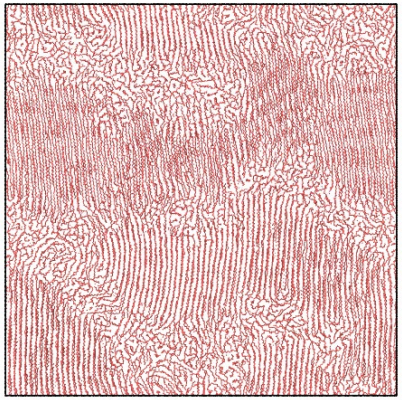
実験との比較による妥当性検証

- 長時間(2000 nsec)の結晶化計算と、一軸延伸

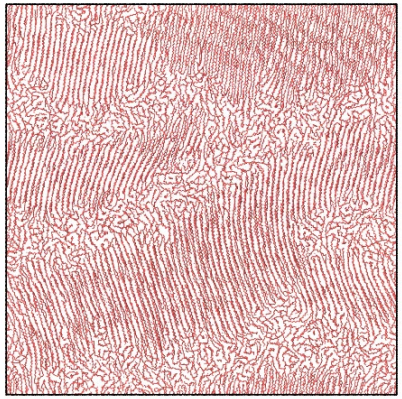
事前の100%延伸



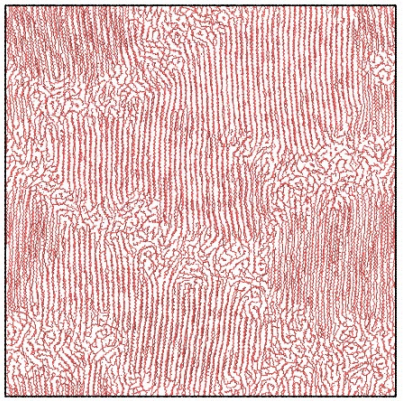
事前の300%延伸



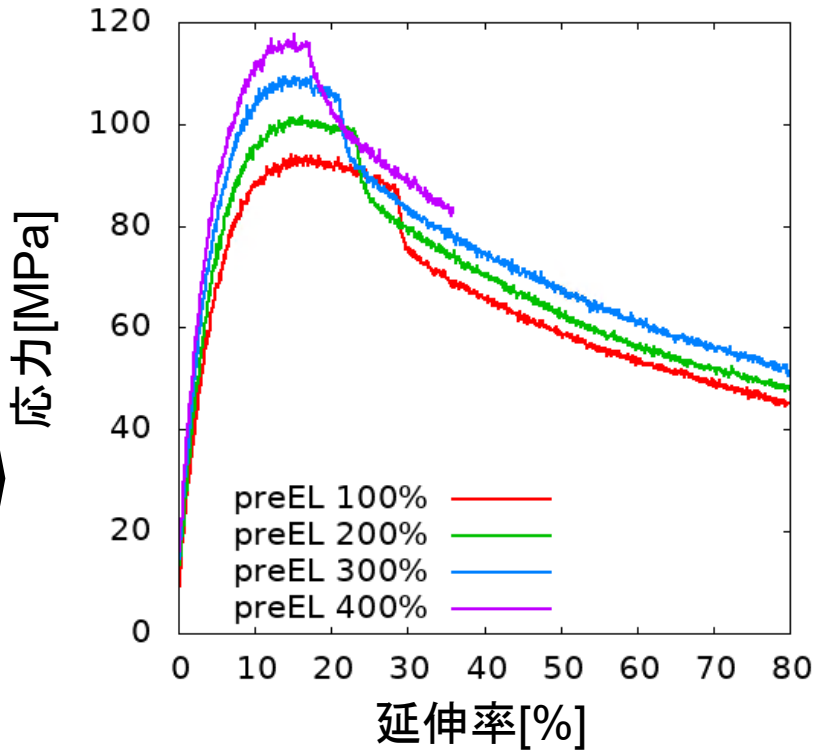
事前の200%延伸



事前の400%延伸



※事前延伸が大きいほど、結晶度大

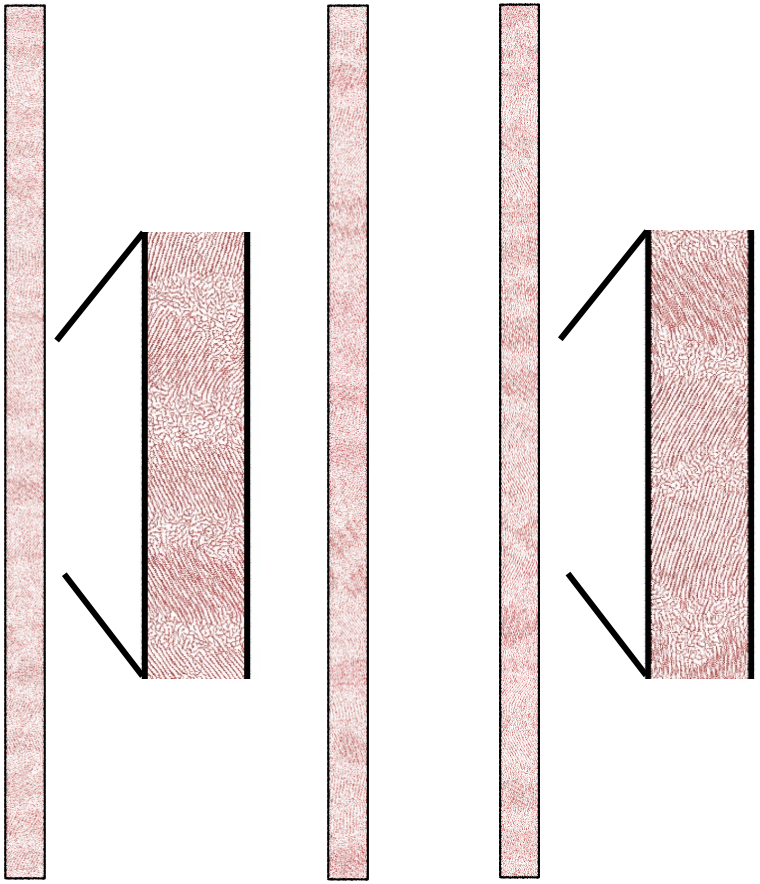


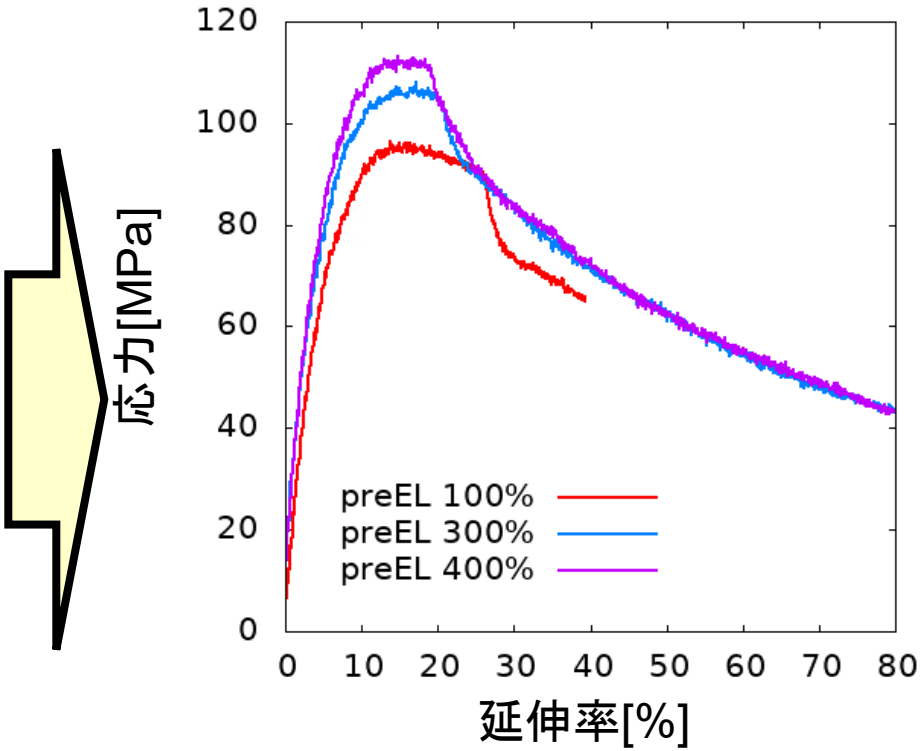
※事前延伸が大きいほど、
応力値の伸びが大きい。
→ 実験の挙動と一致

X線散乱との比較の予備検討

- ミルフィーユ（ラメラ）の繰り返し構造での検討
- 600 nsecの結晶化計算と、一軸延伸

事前100%　事前300%　事前400%



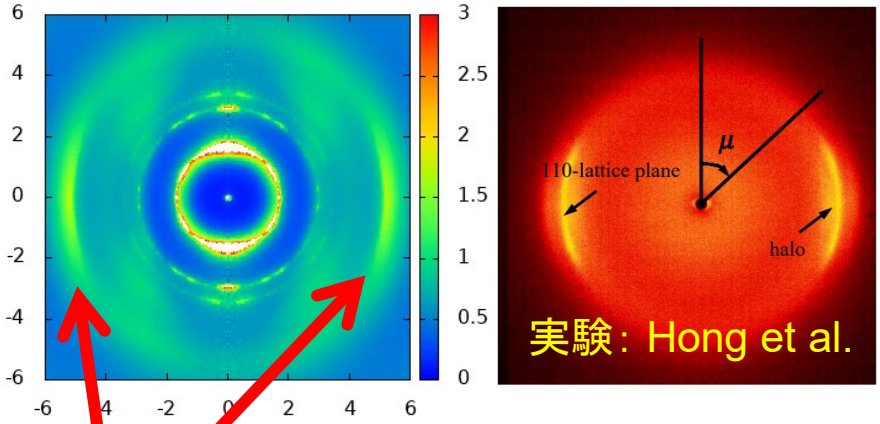


※ほぼ同じ、挙動。

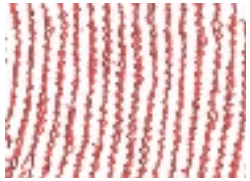
X線散乱との比較の予備検討

- 散乱関数は、フーリエ変換を畳み込んだ量。
- 広角散乱は、原子スケール量。（サイズ依存弱）
- 小角散乱の解像度は、システムサイズに依存。

広角散乱

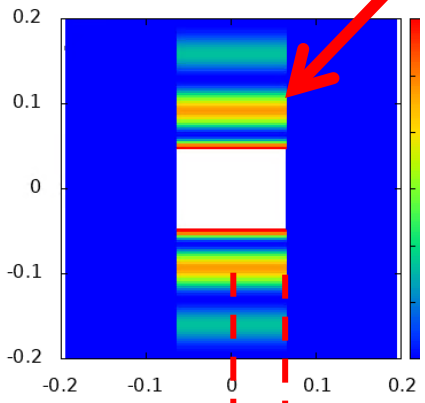


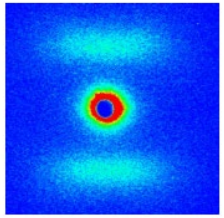
結晶に由来したピーク



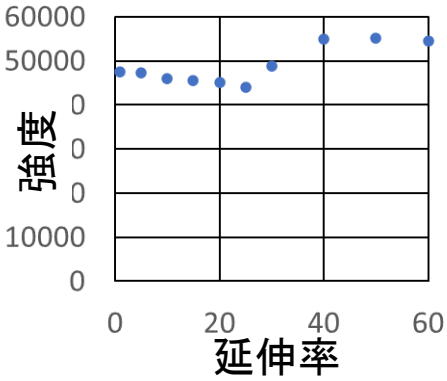
小角散乱

ラメラ（ミルフィーユ）の間隔に由来したピーク（密度差を反映）





$\Delta q_x = 2\pi / L_{pbc,x}$

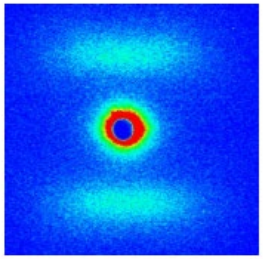


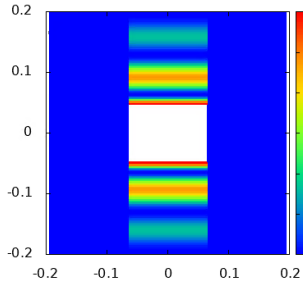
※実験と一致してそう。

まとめと、課題点

- 名大「不老」GPUの集中的利用で、短期間に、結果を得ることができたが、解像度が悪い。

⇨ 8～50倍程度の超大規模系なら、綺麗に比較できそう。

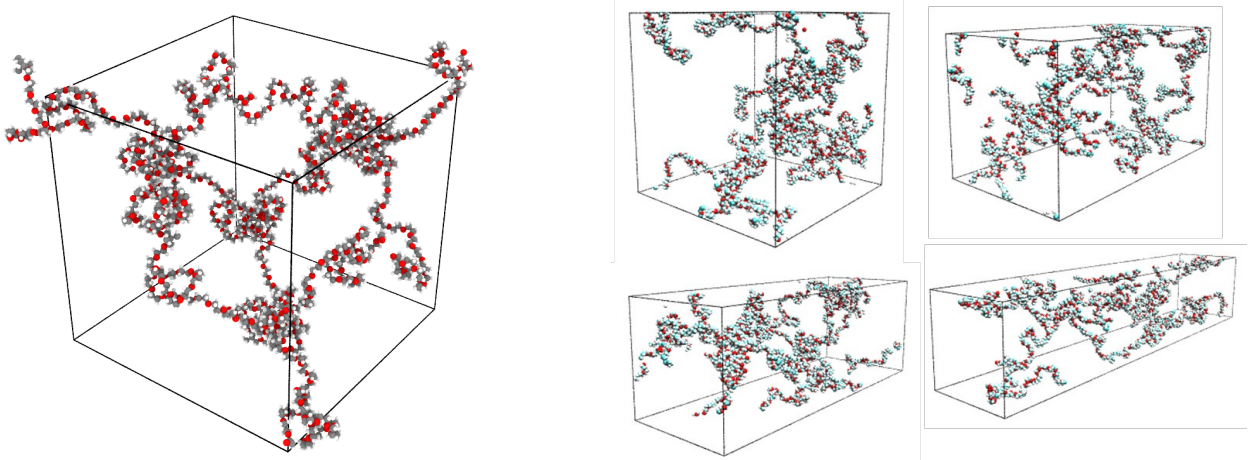




- このテーマで「富岳」に挑戦するか？（or5年待つか）
- 東大・阪大の試験利用で、挑戦的計算を実施。
  - 1600万粒子系を計算済。（A100x8 x8ノード）
    - 12個の条件で、700nsの計算を実施。
  - 現在、一軸延伸で応力歪み曲線と2次元散乱パターンを計算中。

GromacsでのGPU計算の状況

- 水中PEGのMD計算　160,328原子（ほとんど水）



- このサイズで、クーロン力がある場合は、thread-based MPI のsingle GPUが速い。

（阪大SQUID）

（ns/day）	1GPU	2GPU	4GPU	6GPU	8GPU
Thread並列	**32.863**	19.169	23.158	21.978	22.874
MPI	—	7.558	2.370	1.125	0.359

- MPIでは、みるみる遅くなる！　PMEが悪化。
- 1ノードに複数GPUがあるので、無駄なく使いたい。

Single-GPUの複数計算実行

- PythonのsubprocessでしのぐのがⅠ簡単。

```
#PBS
python3 ./run-1.py
```

```
from concurrent import futures
import time
import sys
import os
import subprocess

num_tasks=4
num_workers=4

def exec_func(index):
print('index: %s started.' % index)
    shcmd='cd '+str(index)+' ; '+ ./run.sh '+str(index)+' >log-'+str(index)
    process = subprocess.run(shcmd, stdout=subprocess.PIPE,shell=True)
    time.sleep(5)
    print('index: %s ended.' % index)

future_list = []
with futures.ThreadPoolExecutor(max_workers=num_workers) as executor:
    for i in range(num_tasks):
        future = executor.submit(fn=exec_func, index=i)
        future_list.append(future)

    _ = futures.as_completed(fs=future_list)

print('completed.')
```

ここから、
CUDA_VISIBLE_DEVICESを指定して、
Gromacsでも、何でも呼べばOK。

LAMMPSでのGPU計算の状況

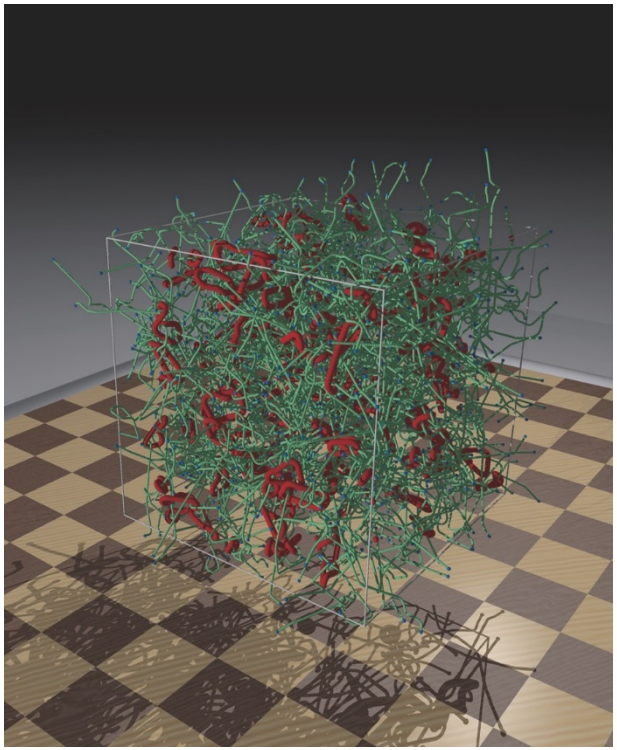
- ReaxFF計算
  - KOKKOS 1000 steps 2,149,216 atoms

ノード数	MPI並列	東大BDEC (sec)	GPU数
1	4(oMPI)	255.456	4
1	6(oMPI)	181.055	6
1	8(oMPI)	144.520	8
2	16(oMPI)	71.499	16
4	32(oMPI)	39.472	32
8	64(oMPI)	26.763	64

GPUを駆使した計算の研究成果

- hp200048（HPCI）、hp200168（富岳）で準備。GPUでプロダクト計算。
  - 環状鎖と線状鎖が混合した系の一連の計算

- K. Hagita, T. Murashima, Effect of Chain-Penetration on Ring Shape for Mixtures of Rings and Linear Polymers. *Polymer* **2021**, 218 123493.
- K. Hagita, T. Murashima, Multi-Ring Configurations and Penetration of Linear Chains into Rings on Bonded Ring Systems and Polycatenanes in Linear Chain Matrices. *Polymer* **2021**, 223 123705.
- K. Hagita, T. Murashima, Molecular Dynamics Simulations of Ring Shapes on a Ring Fraction in Ring–Linear Polymer Blends. *Macromolecules* **2021**, in press.
- T. Murashima, K. Hagita, T. Kawakatsu, Viscosity Overshoot in Biaxial Elongational Flow: Coarse-Grained Molecular Dynamics Simulation of Ring-Linear Polymer Mixtures. *Macromolecules* **2021**, 54, 7210.



Macromolecules 誌の Supplementary Cover Art

JHPCN課題: jh210035

- サイエンス的には、

鎖の交差禁止をON/OFFできるモデルの高速コード開発
（ON/OFFで、熱力学的状態を維持する条件で）

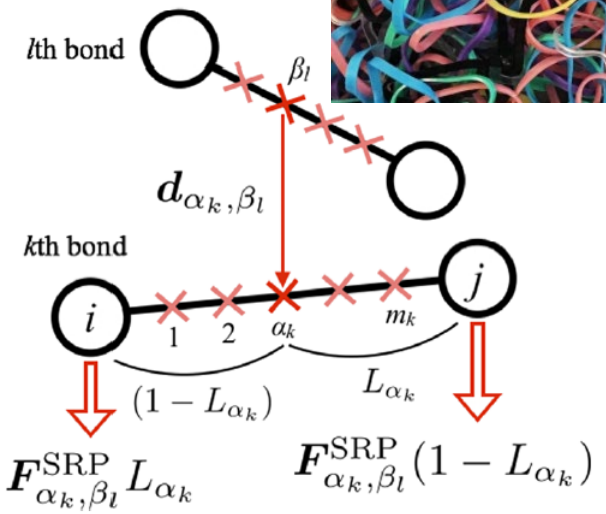
  - 本格計算は、来年度以降を想定。

- 技術的には、
  - CPUコードの最適化の追求
    - コードの一般公開に向けて必要な作業
  - GPU（A100, NVswitch）の活用
    - GPU演算の最適化
    - GPU Direct (NVswitch) の活用
    - GPUのシステム側の機能の理解と活用
    - OpenACCの効率的な利用方法

JHPCN課題でGPUコード開発中

- 鎖の交差の禁止：　環状鎖の問題
  - 交差のON/OFFで物性が変わる。

- 計算モデルは？
  - DPD法のMP-SRP法で可能。
    - 多数の斥力中間点を挿入。
    - 但し、計算コスト大
      → GPUで解決を期待。



  - 計算特性は？
    - 4粒子間の力の計算の問題になる。
      - 正確には、2粒子間と、その隣の2粒子の合わせて4粒子。
    - 斥力中間点の計算は、SIMD向き。
    - 幾何学的条件で、計算不要を判定可能。

最新GPU利用技術の獲得状況

- Multi-nodes, Multi-GPUs
  - Gromacsのマルチノード利用 ※知見をHPCI課題で活用
    - 名大 V100
    - 東大 A100
- Single-node, Multi-GPUs
  - OpenACCプログラミング
    - 東大 A100
  - TensorFlowのノード内並列学習
    - 東大 A100
  - LAMMPSのKokkos(GPU)活用
    - 阪大 A100 (+東北大金研 V100)

東大（講習会）
[6/22,29] 第3回GPUミニキャンプ
[6/23,30] 第4回GPUミニキャンプ

まとめ

- たくさんのGPU（V100, A100）を使うことで、たくさん研究が推進できています。　*感謝感謝*

- No GPU, No research-Life！

- 困っていること。
  - Cuda-aware MPIのucxが、謎！！
  - GPU間の通信の把握が、結局、謎感大！

  - Cuda-aware-MPIの使い方の自動チューニングのツール（フレームワーク？）があると嬉しい。。。
    - 現状、dictionaryもよく分からん感じ。

GPU内の通信の把握が難解？？

- nsysやnvidia-smiを見るのが大変だああ。。。
- 対応すべき策もよく分からないなああ。。。。。
  - CuFFTの2D-FFTで、マルチGPUの性能評価の例

nvidia-smi

8GPU	ns	KB		
PtoP	440,092,112	33,030,144.00	0.075052797	75.05279713
HtoD	389,207,667	9,437,184.00	0.024247169	24.2471688
DtoH	374,914,079	9,437,184.02	0.025171591	25.17159144
DtoD	12,406,587	4,718,592.00	0.380329578	380.3295782

nsys

Time(%)	Total Time (ns)	Operations	Average	Minimum	Maximum	Operation
36.2	440,092,112	1,022	430,618.5	54,816	21,225,036	[CUDA memcpy PtoP]
32.0	389,207,667	576	675,707.8	667,702	681,781	[CUDA memcpy HtoD]
30.8	374,914,079	583	643,077.3	2,496	823,348	[CUDA memcpy DtoH]
1.0	12,406,587	146	84,976.6	30,143	1,594,536	[CUDA memcpy DtoD]
0.0	13,888	5	2,777.6	2,432	3,424	[CUDA memset]

Total	Operations	Average	Minimum	Maximum	Operation
9,437,184.020	583	16,187.280	0.004	16,383.938	[CUDA memcpy DtoH]
9,437,184.000	576	16,384.000	16,384.000	16,384.000	[CUDA memcpy HtoD]
4,718,592.000	146	32,319.123	12,288.000	786,432.000	[CUDA memcpy DtoD]
33,030,144.000	1,022	32,319.123	12,288.000	786,432.000	[CUDA memcpy PtoP]
0.020	5	0.004	0.004	0.004	[CUDA memset]