Süper Bilgisayarlarda Benchmarking ve İş Yükü Ölçeklendirme Nasıl Yapılır?

VASP ve Quantum Espresso

Hande Toffoli
Arda Erbasan

Orta Doğu Teknik Üniversitesi, Fizik Bölümü

Gerekli DFT bilgisi

- Yük yoğunluğu fonksiyoneli teorisi (DFT) çok parçacıklı Schrödinger denkleminin çözülmesini mümkün kılan bir hesaplamalı yöntemdir.
- DFT, odağı dalga fonksiyonundan yük yoğunluğuna çevirir.
- Çok parçacıklı Schrödinger denklemi şu şekilde yazılır:

$$\hat{H}\Psi(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = E\Psi(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)$$

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{N_e}\nabla_i^2 + \frac{1}{2}\sum_{i\neq j}^{N_e}\frac{1}{|\vec{r}_i - \vec{r}_j|} + \sum_{i=1}^{N_e}\sum_{I=1}^{N_n}\frac{Z_I}{|\vec{r}_i - \vec{R}_I|}$$

i,j: Elektron indisleri
I,J: Atom indisleri

Gerekli DFT bilgisi

- Elektronik enerji, yük yoğunluğu cinsinden yazılır.
- Kohn-Sham orbitalleri cinsinden yük yoğunluğu:

$$n(\vec{r}) = \sum_i |\phi_i(\vec{r})|^2$$

- Bu durumda Schrödinger denklemi yerine geçen Kohn-Sham denklemleri şu şekilde yazılır:

$$\left[-\frac{1}{2}\nabla^2 + V_{\text{ext}}(\vec{r}) + \int \frac{n(\vec{r}')}{|\vec{r}-\vec{r}'|}\, d\vec{r}' + V_{\text{xc}}(\vec{r})\right] \phi_i(\vec{r}) = \epsilon_i \phi_i(\vec{r})$$

V_{ext}: İyonik potansiyel

V_{xc}: Değiş-tokuş/korelasyon potansiyeli

Gerekli DFT bilgisi

- Bu denklem sistemini gerçek uzayda çözmek zor olduğu için Kohn-Sham orbitalleri düzlem dalgalar cinsinden açılırlar.

$$\phi_i(\vec{r}) = \sum_{\vec{k}} c_{i\vec{k}} e^{i\vec{k}\cdot\vec{r}}$$

- Bloch teoremi kullanılarak yeniden yazılan Kohn-Sham orbitalleri

$$\Psi_{\vec{k}}(\vec{r}) = \frac{1}{\Omega} \sum_{\vec{G}} u_{\vec{k}}(\vec{G}) e^{i(\vec{k}+\vec{G})\cdot\vec{r}}$$

- Burada **k** ve **G** vektörleri sırasıyla Fourier açılımında kullanılan k noktaları ve ters örgü (reciprocal lattice) vektörleridir.

Gerekli DFT bilgisi

- Bu açılımla Kohn-Sham denklemleri basit bir lineer cebir problemine dönüşür.

$$\Psi_{\vec{k}}(\vec{r}) = \frac{1}{\Omega}\sum_{\vec{G}} u_{\vec{k}}(\vec{G}) e^{i(\vec{k}+\vec{G})\cdot\vec{r}}$$

- Her **k** için Kohn-Sham denklemleri bir kez çözülür.

$$\sum_m H_{m'm}(\vec{k}) c_{i,m}(\vec{k}) = \epsilon_i(\vec{k}) c_{i,m'}(\vec{k})$$

$$H_{m'm}(\vec{k}) = \frac{1}{2}|\vec{k} + \vec{G}_m|^2 \delta_{m'm} + V_{eff}(\vec{G}_m - \vec{G}_{m'})$$

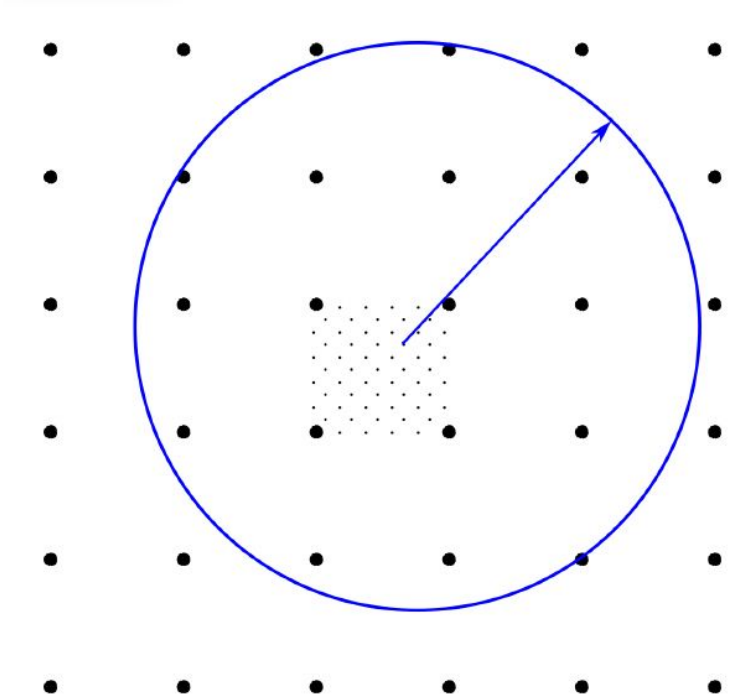
Gerekli DFT bilgisi

- Küçük noktalar: **k** noktaları
- Büyük noktalar: **G** vektörleri
- **k** noktaları Monkhorst-Pack örgüsünden

$$\vec{k}_{n_1,n_2,n_3} = \sum_i^3 \frac{2n_i - N_i - 1}{2N_i} \vec{b}_i$$

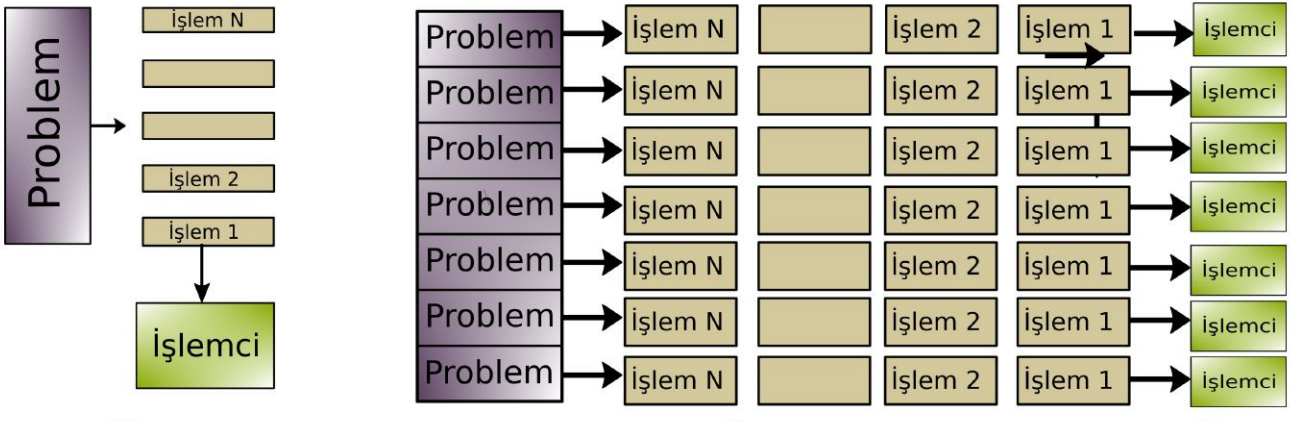
- **G** noktalarının sayısı eşik enerjisinden

$$\frac{1}{2}\left|\vec{k} + \vec{G}\right|^2 \leq E_{cut}$$



Paralleştirme

- Bir kodun paralelleştirilmesi, yapması gereken hesapların daha küçük parçalara bölünüp, birçok işlemci üzerinde aynı anda yapılmasıdır.
- Basit bir örnek olarak iki vektör toplamı verilebilir.

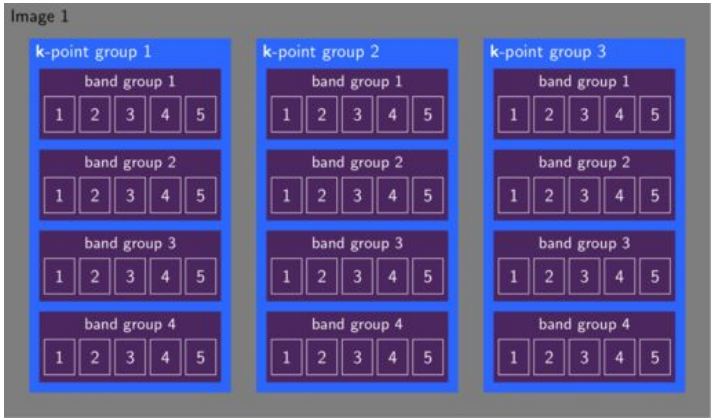


$$\vec{A}+\vec{B}=\sum_{i=1}^{N}(A_i+B_i)=\underbrace{\sum_{i=1}^{n}(A_i+B_i)}_{C1}+\underbrace{\sum_{i=n+1}^{2n}(A_i+B_i)}_{C2}+\cdots+\underbrace{\sum_{i=N-n+1}^{N}(A_i+B_i)}_{Cn}$$

DFT kodlarında paralelizasyon kanalları

- Paralelleştirme neden gerekli?
  - Yapılacak işlemlerin hacmi ve süresi çok uzun olduğu için iş yükünü dağıtma gerekliliği doğar.
- Hangi türde paralelleştirme yapılır?
  - **MPI (Message-Passing Interface):** İşin küçük parçalarının birçok işlemcide aynı anda yapılması
  - **OpenMP:** Bir işlemcide çeşitli *thread*'ler kullanarak ekstra bir paralelleştirme yapılması

- Periodik DFT kodlarında genelde en az üç seviyede paralelleştirme mevcuttur:
  - Bandlar üzerinden
  - k noktaları üzerinden
  - Fourier Transform gridi üzerinden (FFT)



DFT kodlarında paralelizasyon kanalları

- VASP (INCAR dosyasında)

```
LREAL = Auto
NCORE = 55
LSCALAPACK = .FALSE.
KPAR = 1
```

- QE (mpirun komutu opsiyonları) : -nimage, -npools, -nband, -ntg, -ndiag

```
mpirun -np 4096 ./neb.x -ni
8 -nk 2 -nt 4 -nd 144 -i
my.input
```

DFT kodlarında paralelleştirme

- Bandlar üzerinden paralelleştirme:
  - Verilen bir k-noktası için mevcut olan tüm Kohn-Sham orbitallerinin işlemciler üzerinde dağıtılması şeklinde gerçekleşir - veri paralelleştirmesi
  - Bandlar sistemdeki valans elektronları sayısı (spin açık olan sistemlerde yarısı) şeklinde tanımlanır.
  - Metallerde hesaba otomatik olarak ekstra bandlar da eklenebilir.
- Her işlemciye az sayıda (2-8 arası) band gelmesi genelde tercih edilen dağılımdır.
- VASP kodundaki anahtar kelime
  - NBANDS
- QE kodundaki anahtar kelime
  - -nb

DFT kodlarında paralelleştirme

- k-noktaları üzerinden paralelleştirme
  - Monkhorst-Pack örgüsü ile alınan k-noktalarının işlemcilere dağıtılması şeklinde gerçekleşir.
- İdeali, yeteri kadar hafıza varsa, her işlemciye bir tane k-noktası düşmesidir.
  - Pratikte, azaltılamaz (irreducible) k-noktası sayısının bir böleni olarak alınmasında yarar vardır.
- VASP kodunda anahtar kelime
  - KPAR: Kaç tane k-noktası grubu olacak?
  - Örnek: Eğer 24 adet indirgenemez k-noktası varsa ve KPAR 12 ise, her grupta 2 adet knoktası vardır.
- QE kodunda anahtar kelime
  - -nk

DFT kodlarında paralelleştirme

- Fast Fourier Transformu (FFT) örgüsü üzerinden paralelleştirme
  - k uzayı ve gerçek uzay arasındaki transformun işlemcilere bölünmesidir.
- k gruplarını band gruplarına ayırır ve her band grubu için transformun düzlem dalga katsayılarını belli sayıda işlemciye böler.
- VASP kodunda anahtar kelime
  - NCORE: Elimizde olan işlemci sayısının bir tam böleni
  - Birden fazla düğüm kullanıldığı zaman ya da düğümler arasındaki veri aktarımı yavaş olduğunda daha etkili olur.
- QE kodunda anahtar kelime
  - -nt

DFT kodlarında paralelleştirme

Yukarıdaki paralleşme seviyeleri dışında başka seviyeler de mevcuttur:

- *Görüntüler* (images) üzerinden paralelleştrime
  - *Nudged elastic bands* (NEB) benzeri, tamamen birbirinden bağımsız hesapların ayrı işlemci gruplarında yapılmasıdır.
- Lineer Cebir Paralelleştirmesi
  - Kohn-Sham denklemleri lineer cebir denklemleri şeklinde yazıldıktan sonra, matris denklemleri de paralelleştirilebilir.
  - Örn: LSCALAPACK (VASP), -nd (QE)
- Elektronik optimizasyon paralelleşmesi
  - VASP’ta NSIM parametresi optimizasyon sırasında bir seferinde kaç tane k noktasının optimize edileceğini belirler.

Öneriler

- Genelde en verimli hesaplar birden fazla paralelleşme seviyesini birleştirerek elde edilir.
- Sonuç itibarıyla toplam *rank* sayısı

$$N = \text{bandları paralelleştiren rank'ler} \times NCORE \times KPAR$$

şeklinde belirlenir.

- Buradaki parametrelerin seçimi hesabın hızını etkiler.
- Her mimari ve her sistem için ayrı bir parametre tablosu uygun olacaktır.
- Çok sayıda ve uzun hesaplar yapmadan önce muhakkak gerçekte kullanacağınız her sistem için bir iş dağıtım ön çalışması yapmakta yarar vardır.

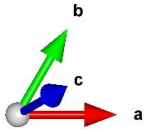
Öneriler

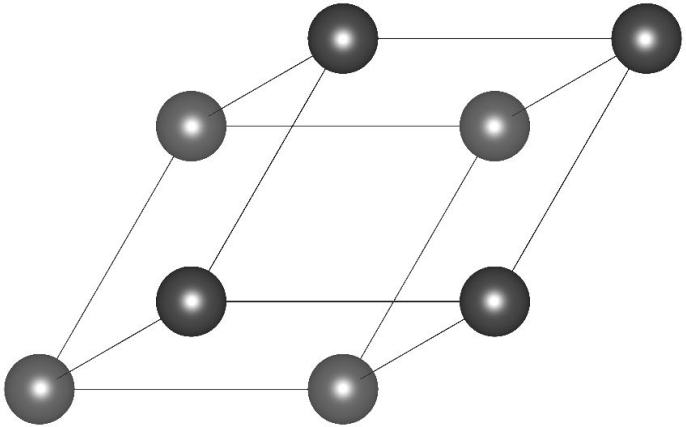
- Ön hesabı yaptıktan sonra (tek bir elektronik adım bile olabilir) hangi alt kısımlarda en çok zaman harcadığının dökümüne bakmak faydalı olur.

```
   POTLOK:  cpu time      0.0215: real time      0.0232
   SETDIJ:  cpu time      0.0019: real time      0.0020
   EDDIAG:  cpu time     32.8441: real time     33.0087
 RMM-DIIS:  cpu time     98.3829: real time     98.9760
   ORTHCH:  cpu time      3.6426: real time      3.6644
      DOS:  cpu time      0.1190: real time      0.1200
   CHARGE:  cpu time     10.4392: real time     10.4827
   MIXING:  cpu time      0.0010: real time      0.0010
     LOOP:  cpu time    145.4523: real time    146.2781
```

Örnek 1: TRUBA ARF kümesinde FFC Si

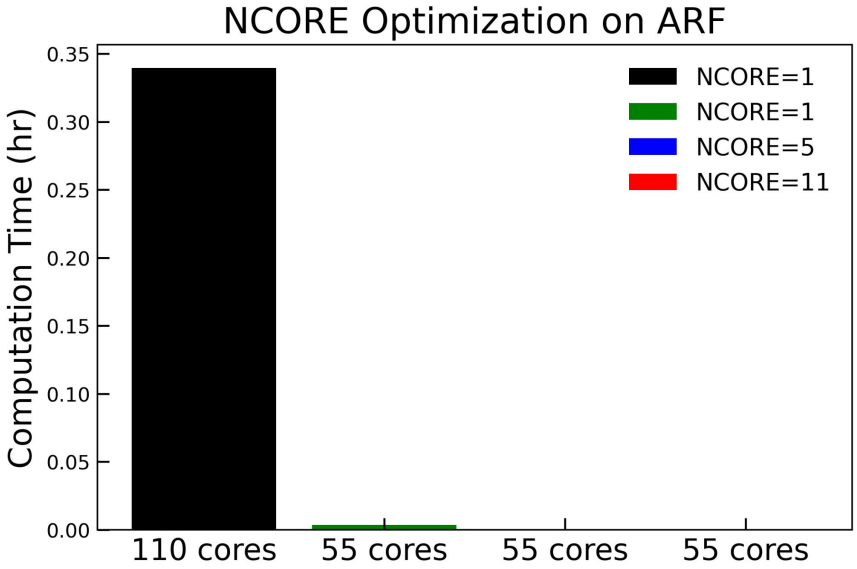
- FCC Si - birim hücrede dört atom
- MP örgüsü üzerinde k noktaları: 11 x 11 x 11
- ENCUT: 520 eV

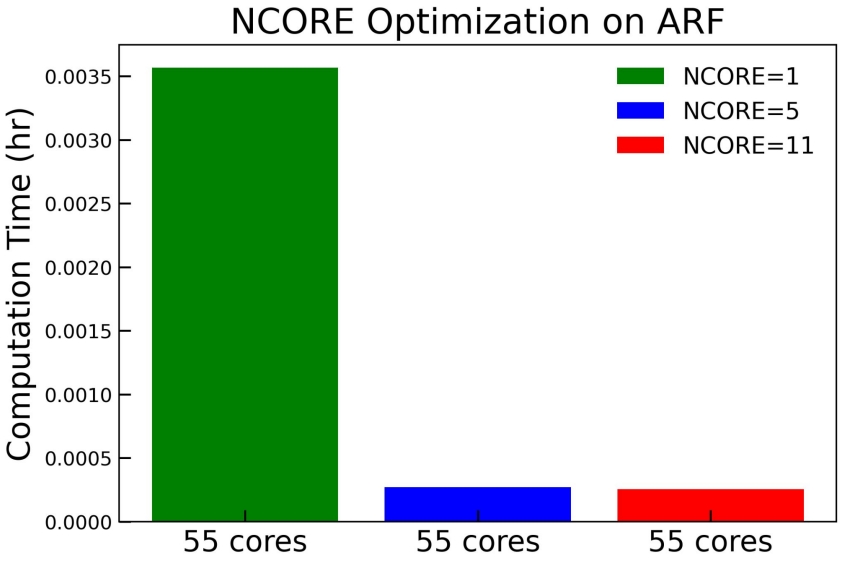




Örnek 1: TRUBA ARF kümesinde FFC Si

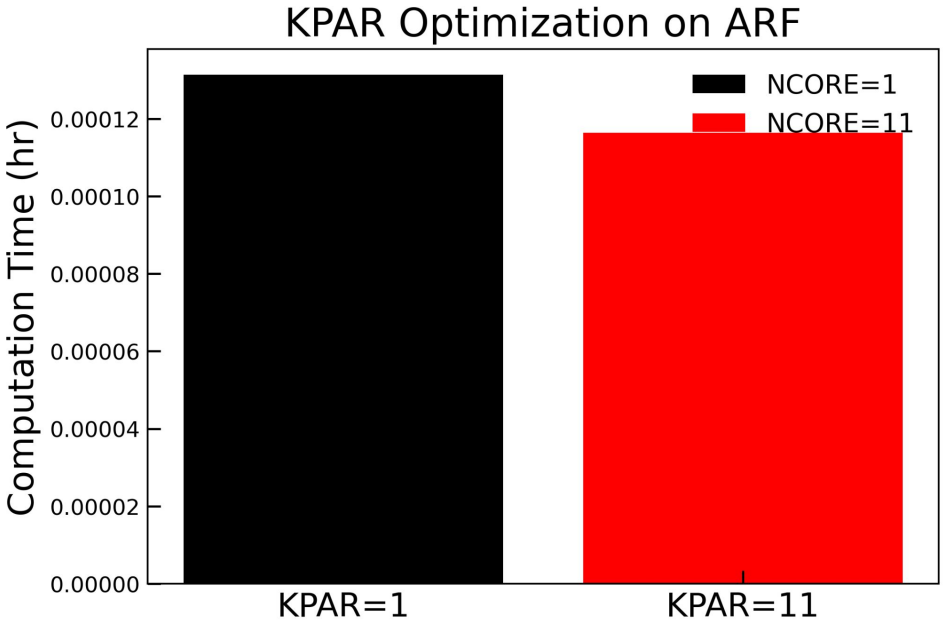
- NCORE paralelleştirmesi
- Kullanılan orfoz kümesindeki düğümlerde ikişer thread'li 56 işlemci mevcuttur.
- Ancak sadece 55 tanesi kullanılabilir (2 tanesi I/O için ayrılmış).





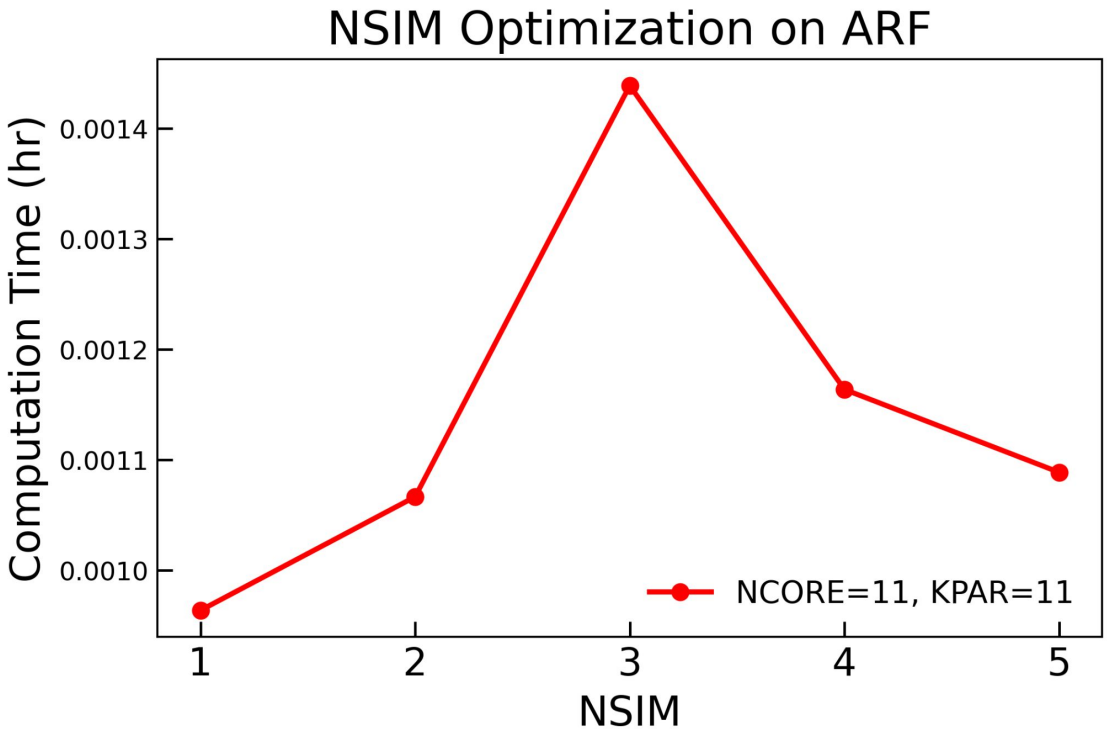
Örnek 1: TRUBA ARF kümesinde FFC Si

- KPAR paralelleştirmesi
- NCORE=55, büyük ihtimalle hafıza probleminden dolayı, hata vermiştir.

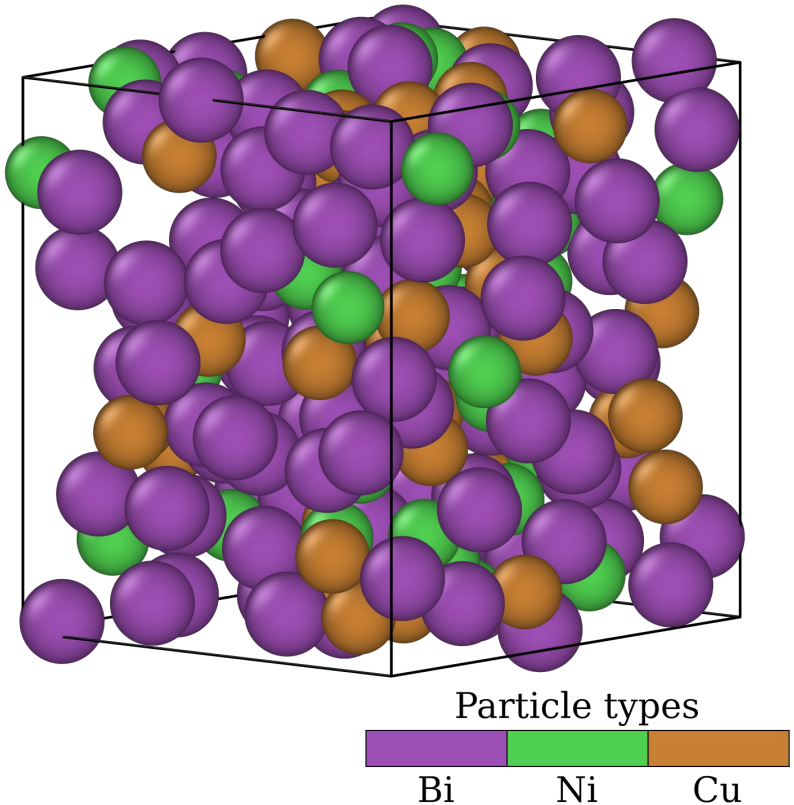


Örnek 1: TRUBA ARF kümesinde FFC Si

- NSIM optimizasyonu



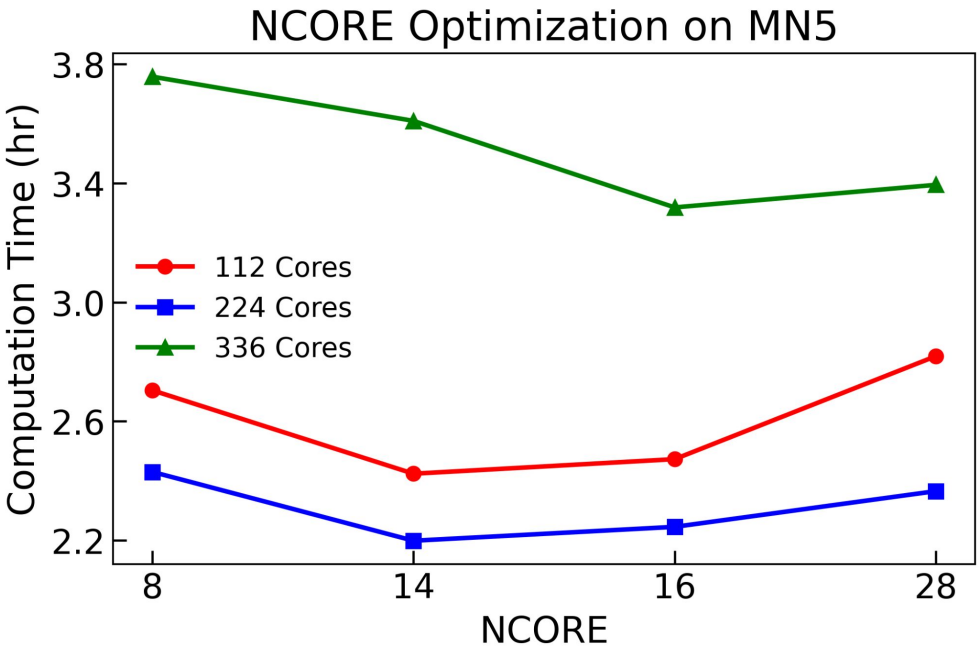
Örnek 2: MareNostrum 5’te likit metal alaşımı



- Bi, Ni, Cu alaşımı
- 200 atom
- MP örgüsü: 2x2x2
- MN5 sistemi: 112 işlemcili sistem
- ENCUT: 520 eV

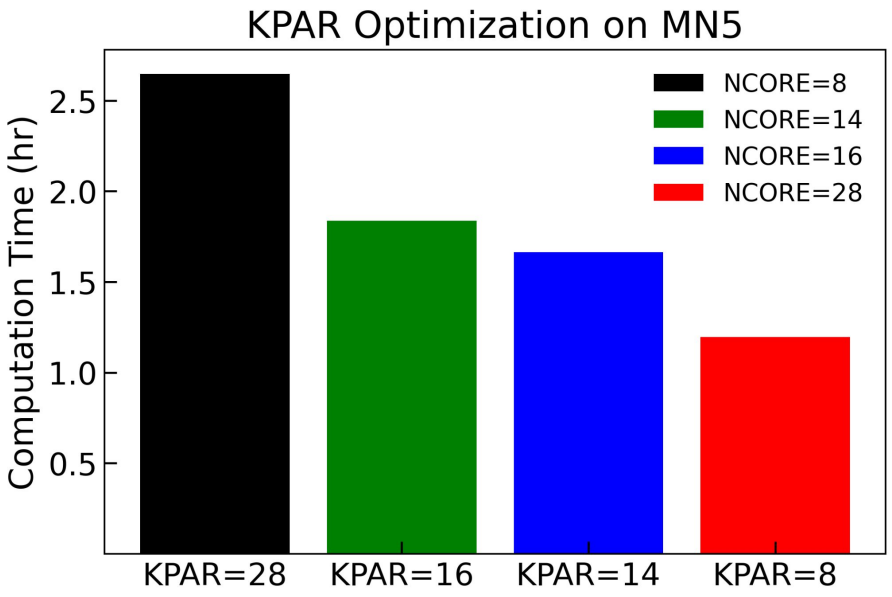
Örnek 2: MareNostrum 5’te likit metal alaşımı

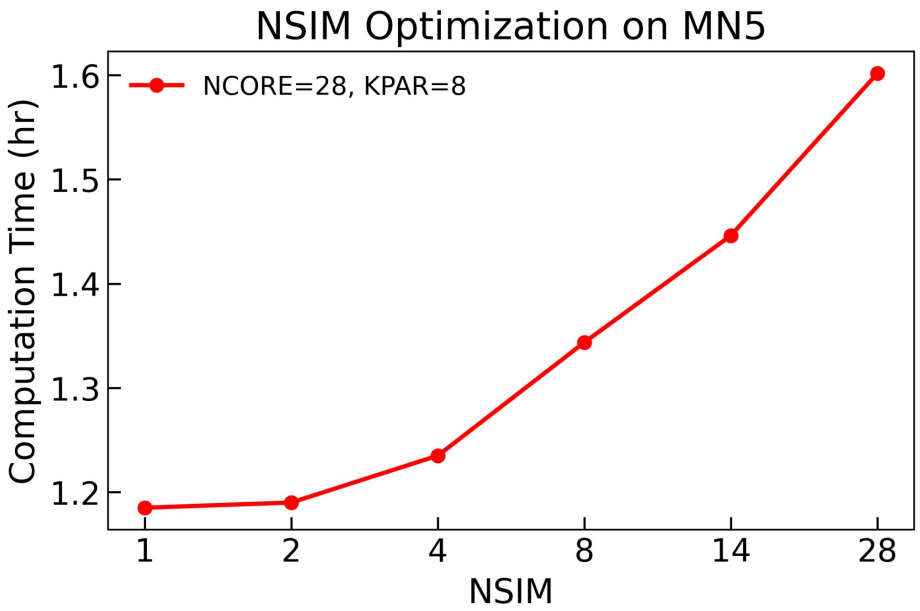
- NCORE paralleştirmesi
- 2 düğüm (node), 1 ve 3’ten daha hızlı çalışmaktadır.
- Bunun sebebi paralleşmeden kazanılan hızlanmanın düğümler arasındaki iletişime kaybedilmesi olabilir.
- NCORE’un da optimal bir değeri bulunmaktadır.



Örnek 2: MareNostrum 5’te likit metal alaşımı

- KPAR ve NSIM optimizasyonu





Kaynaklar

- https://www.vasp.at/wiki/index.php/Optimizing_the_parallelization
- https://www.vasp.at/wiki/index.php/Category:Parallelization
- https://www.quantum-espresso.org/Doc/user_guide/node20.html