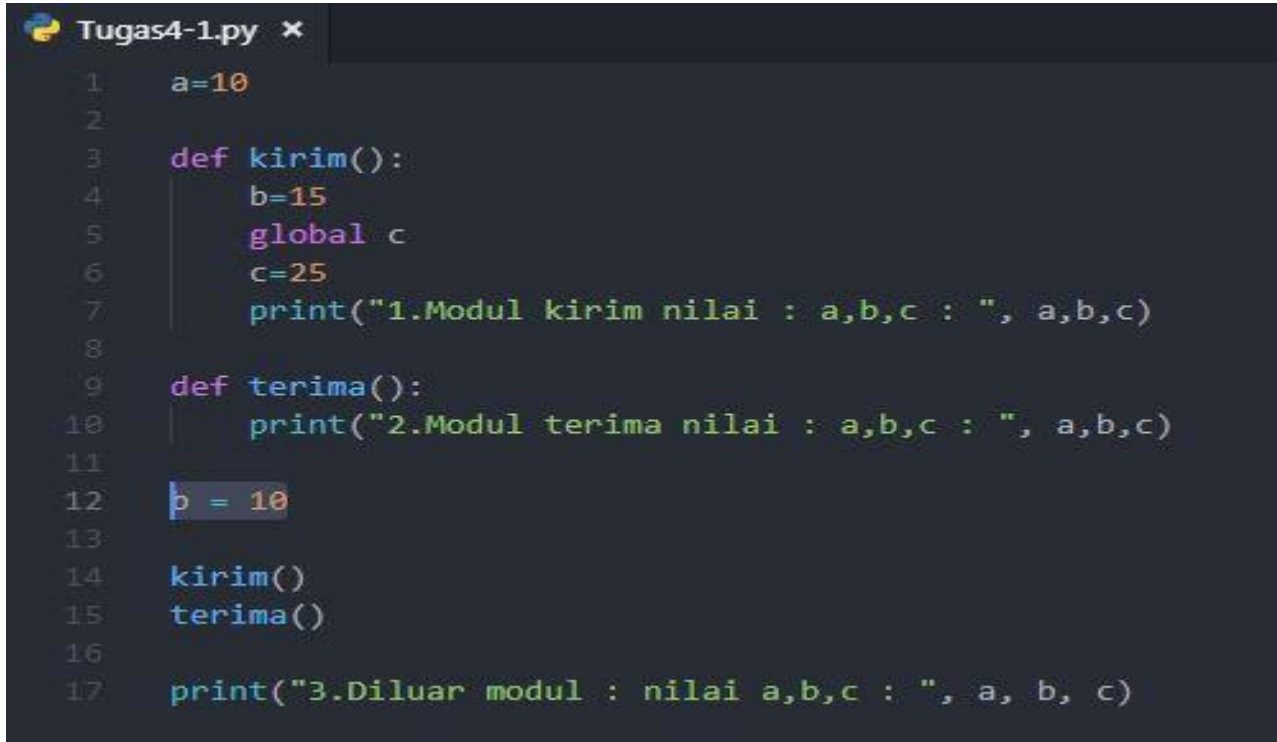


NAMA	INGWE IANNICO
NBI	1461600214
KELAS	R
MATA KULIAH	KOMPUTASI PARALEL

JAWABAN NO 1.

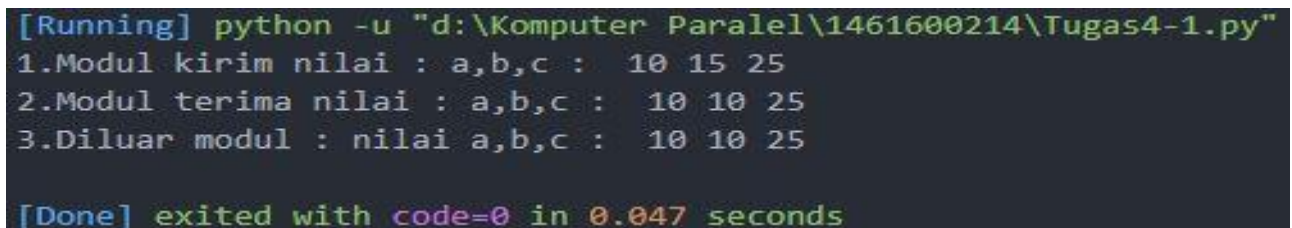


```

1  a=10
2
3  def kirim():
4      b=15
5      global c
6      c=25
7      print("1.Modul kirim nilai : a,b,c : ", a,b,c)
8
9  def terima():
10     print("2.Modul terima nilai : a,b,c : ", a,b,c)
11
12  b = 10
13
14  kirim()
15  terima()
16
17  print("3.Diluar modul : nilai a,b,c : ", a, b, c)

```

Gambar 1.1



```

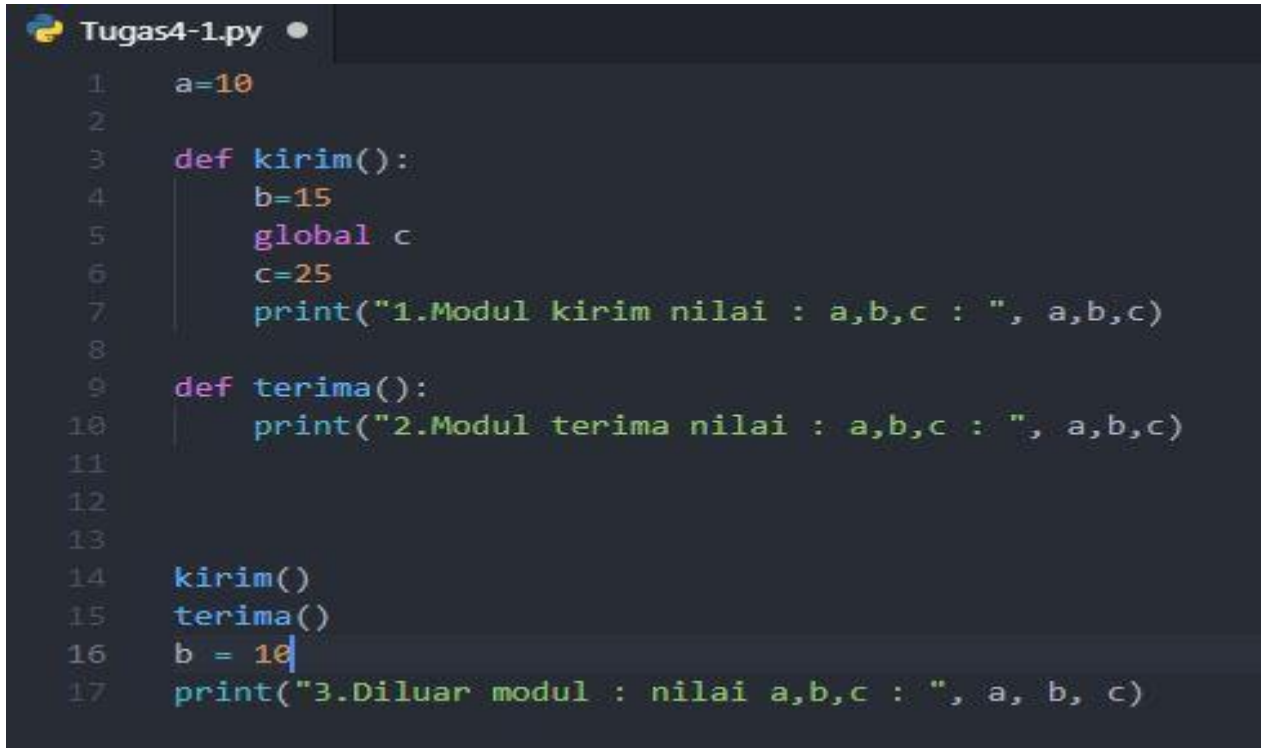
[Running] python -u "d:\Komputer Paralel\1461600214\Tugas4-1.py"
1.Modul kirim nilai : a,b,c :  10 15 25
2.Modul terima nilai : a,b,c :  10 10 25
3.Diluar modul : nilai a,b,c :  10 10 25
[Done] exited with code=0 in 0.047 seconds

```

Output 1.1

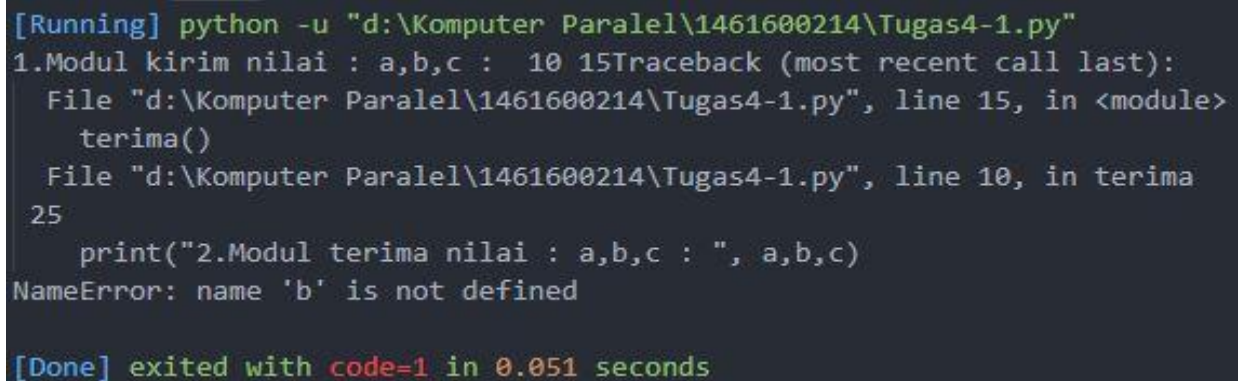
Kesimpulan dari gambar 1.1 dan output 1.1 variable B hanya mengalami perubahan hanya di modul kirim karena variable nya tidak di kirim ke modul terima karena tidak ada koneksi yang menghubungkan antara modul kirim dan modul penerima sehingga modul penerima hanya menampilkan nilai variable B sesuai nilai yang sudah di deklarasikan secara global, sedangkan

nilai variable C dapat di ambil oleh semua modul yang ada karena variable C di deklarasikan sebagai variable global.



```
Tugas4-1.py
1  a=10
2
3  def kirim():
4      b=15
5      global c
6      c=25
7      print("1.Modul kirim nilai : a,b,c : ", a,b,c)
8
9  def terima():
10     print("2.Modul terima nilai : a,b,c : ", a,b,c)
11
12
13
14  kirim()
15  terima()
16  b = 10
17  print("3.Diluar modul : nilai a,b,c : ", a, b, c)
```

Gambar1.2



```
[Running] python -u "d:\Komputer Paralel\1461600214\Tugas4-1.py"
1.Modul kirim nilai : a,b,c : 10 15
Traceback (most recent call last):
  File "d:\Komputer Paralel\1461600214\Tugas4-1.py", line 15, in <module>
    terima()
  File "d:\Komputer Paralel\1461600214\Tugas4-1.py", line 10, in terima
    25
    print("2.Modul terima nilai : a,b,c : ", a,b,c)
NameError: name 'b' is not defined

[Done] exited with code=1 in 0.051 seconds
```

Output1.2

Kesimpulan dari gambar 1.2 dan output 1.2 terjadi error karena pendeklarasian variable di lakukan sesudah proses fungsi di jalankan sehingga proses fungsi tidak mengetahui adanya variable B.

```

Latihan4-2.py x
1 import multiprocessing
2 from multiprocessing import Process, Pipe
3
4 a=10
5 b=10
6
7 def kirim(keneksi):
8     a=[15,25,35]
9     koneksi.send(a)
10    print("nilai yang di kirim: ",a,b)
11    koneksi.close()
12
13
14 def terima (koneksi):
15     print("nilai yang di terima: ", koneksi.recv(),b)
16
17
18 if __name__ == '__main__':
19     PipaIN, PipaOut = Pipe()
20     ProsesKirim=Process(target=kirim,args=(PipaIN,))
21     ProsesTerima=Process(target=terima,args=(PipaOut,))
22     ProsesKirim.start()
23     ProsesTerima.start()
24     ProsesKirim.join()
25     ProsesTerima.join()

```

Gambar 2.1

```

[Running] python -u "d:\Komputer Paralel\1461600214\Latihan4-2.py"
nilai yang di kirim:  [15, 25, 35] 10
nilai yang di terima:  [15, 25, 35] 10

[Done] exited with code=0 in 0.169 seconds

```

Output 2.2

Kesimpulan dari gambar 2.1 dan output 2.2 nilai variable a berubah walaupun sudah di deklarasikan terlebih dahulu tetapi saat di fungsi kirim nilai nya di rubah dan di kirimkan lewat koneksi sehingga fungsi terima bisa memproses nilai baru dari variable a.

JAWABAN NO 2.

```
Proyek1.1.py x
1 import multiprocessing
2 from multiprocessing import Process, Pipe
3 import random
4
5 def kanan(KoneksiKanan):
6     jkanan=random.randrange(1, 2000)
7     print("Jarak Kanan Mobil = ",jkanan,"cm")
8     KoneksiKanan.send(jkanan)
9     KoneksiKanan.close()
10
11
12 def kiri(KoneksiKiri):
13     jkiri=random.randrange(1, 2000)
14     print("Jarak Kiri Mobil = ",jkiri,"cm")
15     KoneksiKiri.send(jkiri)
16     KoneksiKiri.close()
17
18
19 def depan(KoneksiDepan):
20     jdepan = random.randrange(1, 1000)
21     print("Jarak Depan Mobil = ",jdepan,"cm")
22     KoneksiDepan.send(jdepan)
23     KoneksiDepan.close()
24
25
26 def belakang(KoneksiBelakang):
27     jbelakang = random.randrange(1, 1000)
28     print("Jarak Belakang Mobil = ",jbelakang,"cm")
29     KoneksiBelakang.send(jbelakang)
30     KoneksiBelakang.close()
31
32
33 def MainControl(KoneksiKanan,KoneksiKiri,KoneksiDepan,KoneksiBelakang):
34     BatasKanan=KoneksiKanan.recv()
```

```
Proyek1.1.py x
35     BatasKiri=KoneksiKiri.recv()
36     BatasDepan=KoneksiDepan.recv()
37     BatasBelakang=KoneksiBelakang.recv()
38     print ("\n<=== Kondisi Mobil ===>")
39
40     #Mengatur Sistem jika salah satu jarak 0
41     if (BatasKanan==0 & BatasKiri==0 & BatasDepan==0 & BatasBelakang==0):
42         print("MOBIL TERTABRAK!!!!!!")
43
44     #Mengatur Sistem jika bagian kiri dan kanan mobil terlalu mepet
45     elif ((BatasKiri<=40)&(BatasKanan<=40)):
46         print("Hati Hati Bagian Kiri dan Kanan anda Kurang Dari 40cm")
47         print("\n<=== Solusi ===>")
48         print("JAGA KECEPATAN & JARAK!!!")
49
50     #Mengatur Sistem jika bagian Depan dan Belakang mobil terlalu mepet
51     elif ((BatasDepan<=40)&(BatasBelakang<=40)):
52         print("Hati Hati Bagian Depan dan Belakang anda Kurang Dari 40cm")
53         print("\n<=== Solusi ===>")
54         print("JAGA KECEPATAN & JARAK!!!")
55
56     #Mengatur Sistem jika bagian Kanan,Kiri,Depan,dan Belakang mobil terlalu mepet
57     elif((BatasKiri<=40)&(BatasKanan<=40)&(BatasDepan<=40)&(BatasBelakang<=40)):
58         print("Hati Hati Semua Bagian Mobil Kurang Dari 40cm")
59         print("\n<=== Solusi ===>")
60         print("JAGA KECEPATAN & JARAK!!!")
61
62     #Mengatur Sistem jika Bagian Kanan terlalu mepet
63     elif (BatasKanan<=40):
64         print("Jarak Bagian Kanan Mobil Kurang Dari 40cm")
65         print("\n<=== Solusi ===>")
66         print("TOLONG BELOK KE KIRI!!!")
67
```

```
Proyek1.1.py x
68 #Mengatur Sistem jika Bagian Kiri terlalu mepet
69 elif (BatasKiri<=40):
70     print("Jarak Bagian Kiri Mobil Kurang Dari 40cm")
71     print("\n<=== Solusi ===>")
72     print("TOLONG BELOK KE KANAN!!!")
73
74 #Mengatur Sistem jika Bagian Depan terlalu mepet
75 elif (BatasDepan<=200):
76     print("Jarak Bagian Depan Mobil Kurang Dari 2m")
77     print("\n<=== Solusi ===>")
78     print("REM MOBIL SEGERA!!!")
79
80 #Mengatur Sistem jika Bagian Belakang terlalu mepet
81 elif (BatasBelakang<=200):
82     print("Jarak Bagian Belakang Mobil Kurang Dari 2m")
83     print("\n<=== Solusi ===>")
84     print("Tambah Kecepatan!!!")
85
86 #Mengatur Sistem jika semua Bagian dalam jarak aman
87 else:
88     print("Mobil Dalam Kondisi Aman")
89
90
91 if __name__ == '__main__':
92     KananIn, KananOut = Pipe()
93     KiriIn, KiriOut = Pipe()
94     DepanIn, DepanOut = Pipe()
95     BelakangIn, BelakangOut = Pipe()
96     JarakKanan = Process(target=kanan, args=(KananIn,))
97     JarakKiri = Process(target=kiri, args=(KiriIn,))
98     JarakDepan = Process(target=depan, args=(DepanIn,))
99     JarakBelakang = Process(target=belakang, args=(BelakangIn,))
100     SensorSistem = Process(target=MainControl, args=(KananOut,KiriOut,DepanOut,BelakangOut))
101
```

```
Proyek1.1.py x
102 JarakKanan.start()
103 JarakKiri.start()
104 JarakDepan.start()
105 JarakBelakang.start()
106 SensorSistem.start()
107
108 JarakKanan.join()
109 JarakKiri.join()
110 JarakDepan.join()
111 JarakBelakang.join()
112 SensorSistem.join()
```

```
[Running] python -u "d:\Komputer Paralel\1461600214\Proyek1.1.py"
Jarak Kiri Mobil = 857 cm
Jarak Depan Mobil = 614 cm
Jarak Belakang Mobil = 511 cm
Jarak Kanan Mobil = 1611 cm

<=== Kondisi Mobil ===>
Mobil Dalam Kondisi Aman

[Done] exited with code=0 in 0.29 seconds
```

JAWABAN NO 3.

MODEL MEMORI YANG DIBAGI

adalah objek fisik yang memungkinkan analisis atau pengamatannya eksplorasi objek atau proses nyata lainnya. Model mewakili objek yang dieksplorasi dengan cara yang disederhanakan dengan hanya memperhitungkan fitur-fitur dasarnya. Karena penyederhanaan, model lebih mudah dianalisis daripada objek nyata yang sesuai. model komputer yang memungkinkan studi tentang proses komputasi yang dilakukan di dalam komputer tersebut. Model-model, yang disebut model perhitungan, sangat membantu ketika menganalisis dan merancang algoritma, serta dalam menentukan metrik kinerja yang digunakan untuk evaluasi algoritma

Model perhitungan tidak boleh dikaitkan dengan arsitektur komputer tertentu, atau dengan kelas arsitektur seperti itu. Dengan kata lain, independensi mereka dari perangkat keras sangat penting. Fitur penting lainnya adalah fleksibilitas yang memastikan algoritma yang dikembangkan mengadopsi model-model ini dapat diimplementasikan dan dijalankan komputer dengan arsitektur yang berbeda. Ini sangat penting dalam bidang komputasi paralel di mana keragaman arsitektur tinggi. Sebagai hasil dari keanekaragaman ini beberapa model telah maju. Sayangnya, karena jumlahnya relatif besar persyaratan bahwa model harus memenuhi, sebagian dalam konflik satu sama lain, tidak ada model yang dikembangkan sejauh ini telah menjadi model paralel yang diterima secara umum komputasi. Yang sering digunakan adalah model memori bersama (atau paralel acak model mesin akses, PRAM) dan model jaringan. Mereka sesuai dengan perhitungan paralel yang dilakukan dengan menggunakan memori bersama dan dengan mengirim pesan melalui beberapa jaringan komunikasi. Sebelum membahas model-model ini, kami akan hadir model komputasi berurutan yang mendasari model PRAM

MODEL RAM

Model perhitungan sekuensial yang diterima secara luas adalah mesin dengan acak akses memori (RAM). Model1 terdiri dari prosesor dan memori yang mengandung jumlah sel M_i yang berpotensi tak terbatas untuk $i = 1, 2, 3, \dots$. Di setiap sel memori yang diidentifikasi oleh alamatnya i , nilai terbatas yang dinyatakan dalam biner, mungkin sangat besar, dapat disimpan. Model ini mengasumsikan bahwa waktu untuk membaca (menulis) nilai dari (dalam) sebuah sel M_i adalah konstan dan sama dengan satuan waktu, terlepas dari alamat sel. Konstan ini waktu akses adalah karakteristik untuk akses acak, sebagai lawan dari akses berurutan ke sel-sel kaset di mesin Turing. Prosesor RAM dapat menjalankan instruksi dari beberapa daftar terbatas yang mencakup instruksi yang mirip dengan yang diimplementasikan dalam prosesor modern. Nilainya meningkat oleh 1 setelah pelaksanaan instruksi yang berbeda dari JUMP, JPOS, JZERO

dan JNEG. Dalam hal instruksi ini, register L dapat diatur ke nilai i (the argumen kedua dari suatu instruksi), atau meningkat sebesar 1. Prosesor dikendalikan oleh program RAM, yang merupakan urutan instruksi yang mengimplementasikan algoritma yang diberikan. Program tidak disimpan dalam memori, tetapi dalam unit kontrol model. Ini memastikan bahwa program tidak dapat dimodifikasi selama pelaksanaannya. Saat menganalisis RAM program, mudah untuk mengasumsikan bahwa suatu program terdiri dari sejumlah langkah komputasi, di mana setiap langkah dibagi menjadi tiga fase.

- fase pertama prosesor mengambil operan dari sel memori ke register aritmatika
- fase kedua ia melakukan operasi aritmatika atau logis (Boolean) pada isi register aritmatika, operasi kontrol, operasi I / O dll, dan dalam
- fase ketiga ini menyimpan hasil operasi di sel alamat yang ditunjukkan.

Beberapa fase dari langkah komputasi mungkin kosong. Diasumsikan bahwa waktu eksekusi adalah sama untuk setiap langkah komputasi, dan sama dengan satuan waktu. Nilai yang dihitung dari polinomial muncul di sel M1 Program RAM yang diungkapkan oleh instruksi prosesor dirinci dan ditutup ke bahasa mesin. Akibatnya, mereka nyaris tidak terbaca. Karena itu untuk meningkatkan keterbacaan mereka biasanya disajikan pada tingkat abstraksi yang lebih tinggi.

MODEL JARINGAN

Model jaringan terdiri dari sejumlah prosesor yang beroperasi sesuai dengan model RAM. Untuk bertukar data, prosesor dihubungkan oleh saluran komunikasi dua arah, atau tautan, yang membentuk jaringan interkoneksi tidak ada memori bersama dalam model. Prosesor dapat beroperasi baik secara sinkron atau asinkron, tetapi biasanya mode asinkron diasumsikan. Prosesor berkomunikasi dan mengoordinasikan kegiatan mereka dengan mengirim pesan melalui tautan jaringan interkoneksi. Transfer pesan dilakukan dengan prosedur routing yang berjalan di semua prosesor dan bekerja sama dengan masing-masing

lainnya.

Model jaringan mendefinisikan cara prosesor terhubung satu sama lain, yang disebut topologi jaringan interkoneksi. Ini dapat dijelaskan dengan cara grafik yang simpulnya mewakili prosesor, dan tepinya mewakili tautan dua arah antara prosesor. Grafik dapat menjadi diagram alternatif dari model jaringan. Ada banyak topologi jaringan, yang dianalisis dengan mempertimbangkan berbagai parameter yang memungkinkan kami menilai kesesuaian setiap topologi untuk komputasi paralel. Parameter berikut untuk mengevaluasi topologi dibedakan: diameter, derajat, lebar pembelahan, konektivitas tepi, dan biaya. Diameter jaringan adalah jarak maksimum yang diukur dengan jumlah tautan (tepi) antara dua prosesor atau simpul (simpul) dari suatu jaringan. Jaringan adalah semakin baik semakin kecil diameter yang dimilikinya,

karena dalam kasus terburuk suatu pesan harus dialihkan melalui jumlah tautan sama dengan diameter jaringan. Tingkat jaringan adalah maksimum jumlah tautan tetangga dari suatu simpul dalam suatu jaringan. Jika tingkat ini kecil, maka prosedur komunikasi yang melakukan perutean pesan adalah sederhana dan karenanya lebih efisien, karena sejumlah kecil saluran harus dikendalikan oleh prosedur perutean.

Konektivitas tepi adalah ukuran banyaknya jalur antara simpul jaringan. Ini didefinisikan sebagai minimum sejumlah tepi yang harus gagal — untuk dihapus dari jaringan — agar terputus. Semakin besar konektivitas tepi, semakin baik resistensi jaringan terhadap kerusakan. Selain itu, dalam jaringan dengan konektivitas yang lebih tinggi terdapat lebih sedikit pertentangan proses untuk sumber daya komunikasi.

MESH

Garis k -dimensi yang simpulnya disusun dalam array $P [1..n_1, 1..n_2, \dots, 1..n_k]$ di mana? $K \ i = 1 \ n_i = p$, termasuk dalam kelas jaringan jarang. A vertex $P [i_1, i_2, \dots, i_j, \dots, i_k]$ terhubung ke simpul $P [i_1, i_2, \dots, i_j \pm 1, \dots, i_k]$ untuk $j = 1, 2, \dots, k$ jika simpul tersebut ada. Dengan demikian, hampir setiap titik terhubung oleh tepi (tautan) dengan $2k$ simpul lainnya. menggambarkan diagram satu dimensi mesh di mana setiap simpul kecuali untuk dua simpul di ujung memiliki dua tetangga. Melengkapi jaringan dengan koneksi sampul menghasilkan cincin, atau torus satu dimensi Keuntungan dari jerat satu dimensi adalah Gelar kecil sama dengan dan kemudahan penskalaan, sedangkan kelemahan jaringan besar diameter sama dengan $p - 1$. Angka 2.8a dan 2.8b menggambarkan jerat dua dimensi ($k = 2$) derajat 4 dan diameter $O(\sqrt{p})$. Struktur serupa memiliki mesh tiga dimensi Keuntungan dari jerat dua dan tiga dimensi adalah bahwa mereka memungkinkan pemetaan komputasi yang mudah dilakukan pada struktur reguler. Contohnya termasuk perhitungan aljabar linier yang melibatkan operasi matriks, atau perhitungan dalam grafik komputer, di mana bidang gambar dipartisi menjadi potongan-potongan reguler yang diproses oleh prosesor individu. Partisi serupa digunakan dalam algoritma pemodelan spasial dari berbagai macam fenomena, seperti aliran udara di atmosfer untuk memprediksi cuaca, aliran cairan dalam hidrodinamika, distribusi medan magnet dalam inti transformer, dll.

CUBE

dapat berisi sejumlah simpul pada setiap dimensi. Membatasi jumlah ini untuk dua hasil acube, orhypercube, topologi. Kubus A_k -dimensi terdiri dari $p = 2^k$ for $k \geq 0$ simpul berjumlah $0, 1, \dots, 2^k - 1$. Lebih mudah mewakili angka simpul dalam notasi biner, $w \in \{0, 1\}^k$. Dua simpul dihubungkan oleh tepi, jika string sedikit w sesuai dengan jumlah dari simpul simpul ini berbeda hanya pada satu posisi. Kubus nol dimensi terdiri dari satu simpul tunggal ($p = 2^0 = 1$)