BAB III

LANDASAN TEORI

3.1 Harga Opsi

Opsi adalah instrumen keuangan di mana dua pihak setuju untuk menukarkan aset pada harga atau strike dan tanggal atau maturity sudah ditentukan sebelumnya. Dengan membayar di muka atau yang dikenal sebagai harga atau premium dari opsi, pemegang kontrak memiliki hak, tetapi bukan kewajiban, untuk membeli atau menjual aset pada saat maturity (Ramírez-Espinoza, 2011).

Opsi memiliki istilah-istilah yang penting yang perlu dipahami yaitu exercise (strike) price, expiration date, permi opsi. Exercise (strike) price adalah harga per lembar saham yang menjadi tolak ukur pada saat jatuh tempo. Terdapat 2 exercise price yang berbeda yaitu pada saat calls dan puts. Pada saat calls, exercise price merupakan harga yang harus dibayarkan pemilik opsi beli pada saat jatuh tempo. Pada saat putss, exercise price merupakan harga akan diterima pemilik opsi dari penjual opsi pada saat jatuh tempo. Expiration date merupakan batas waktu di mana opsi dapat dilaksanakan. Premi opsi merupakan harga yang harus dibayarkan oleh pembeli opsi kepada penjual opsi.

Opsi memiliki dua tipe utama, yaitu calls dan puts. Calls merupakan kontrak opsi yang memberi hak tetapi bukan kewajiban untuk membeli underlying asset pada harga yang telah ditentukan sebelum atau saat maturity. Puts merupakan kontrak opsi yang memberi hak tetapi bukan kewajiban untuk menjual underlying

asset pada harga yang telah ditentukan sebelum atau saat maturity. Opsi dapat diklasifikasikan menjadi dua kelompok berdasarkan waktu exercise yaitu opsi european dan opsi american. Model opsi european memiliki aturan hanya dapat di exercise pada waktu maturity. Model opsi american memiliki aturan dapat di exercise kapan saja pada rentang waktu antara waktu pembelian kontrak dan waktu maturity.

Nilai dari opsi didapatkan berdasarkan turunan nilai dari underlying asset, sehingga opsi bersifat derivatif. Berdasarkan hal tersebut kontrak opsi merupakan salah satu "keamanan derivatif" (Crack, 2009). Nilai underlying asset memiliki sifat berbanding lurus dengan nilai calls opsi dan sifat berbanding terbalik dengan nilai puts opsi. Nilai calls opsi naik jika nilai underlying asset naik dan sebaliknya. Nilai puts turun jika underlying asset naik dan sebaliknya.

Opsi memiliki nilai intrinsik yang merupakan nilai ekonomis saat opsi dilaksanakan (Tandelilin, 2010). Nilai intrinsik dipengaruhi oleh harga saham dan strike price. Nilai intrinsik dari opsi dapat dilihat pada Tabel 3.1. Jika harga saham lebih besar daripada strike price maka nilai intrinsik akan bernilai positif, bila nilai sama atau strike price lebih besar maka nilai intrinsik bernilai nol. Secara sederhana nilai intrinsik menunjukkan apakah opsi memberikan keuntungan kepada pembeli opsi atau tidak.

Saat harga saham sama dengan strike price maka dikatakan at the money. Saat harga saham lebih besar daripada strike price maka dikatakan in the money.

Saat harga saham lebih kecil daripada strike price maka dikatakan out of the money. Kondisi at the money dan out the money sama-sama memiliki nilai intrinsik nol.

Tabel 3.1 Nilai Intrinsik Dari Opsi

	Harga Saham > Strike Price	
	Calls option	**Puts option**
Nilai Intrinsik	Positif	Nol
	In the money	Out of the money
	Harga Saham < Strike Price	
	Calls option	**Puts option**
Nilai Intrinsik	Nol	Positif
	Out of the money	In the money
	Harga Saham = Strike Price	
	Calls option	**Puts option**
Nilai Intrinsik	Nol	Nol
	At the money	At the money

3.2 *Volatility*

Nilai dari *underlying asset* dalam menentukan nilai opsi direpresentasikan sebagai *volatility*. *Volatility* menjelaskan pergerakan harga dengan menghitung kecepatan dan ukuran dari perubahan *underlying asset*. *Volatility* dapat diukur dengan standar deviasi atau varians. Nilai volatility bernilai tinggi bila terjadi perubahan naik dan turun dalam interval waktu yang singkat dan bernilai rendah

saat hampir tidak mengalami perubahan (Mubarik & Javid, 2015). Nilai *volatility* di masa depan tersebut menjadi indikator kondisi sentimen pasar saat itu. Nilai *volatility* berbanding lurus dengan nilai opsi, yaitu semakin tinggi nilai *volatility* maka semakin tinggi pula nilai opsi.

3.2.1 *Stochastic Volatility*

Hull, White, Scott, dan Wiggins (1987) memperkenalkan *Stochastic Volatility* (SV) yang merupakan suatu pendekatan untuk mengatasi kekurangan *Black-Scholes* yang mengasumsikan bahwa volatilitas harga aset tidak konstan (Wiggins, 1987).

Berdasarkan permasalahan tersebut SV membiarkan volatilitas harga yang sebagai variabel acak bervariasi dari waktu ke waktu atau dapat dikatakan SV mengacu pada nilai berturut-turut dari variabel acak yang tidak independen sehingga dapat meningkatkan perkiraan dan perhitungan akurasi (Lorig & Sircar, 2015).

Secara matematis, SV dapat dimodelkan sebagai berikut.

$$dX_t = -\frac{1}{2}\sigma^2(Y_t)\,dt + \sigma(Y_t)dW_t^{Q} \quad (5)$$

$$dY_t = \alpha(Y_t)\,dt + \beta(Y_t)dB_t^{Q} \quad (6)$$

$$d(W^{Q}, B^{Q})_t = \rho dt \quad (7)$$

3.3 *Heston Model*

Heston model merupakan permodelan harga opsi yang populer dengan permodelan *stochastic volatility*. Model ini ditemukan oleh Steve L. Heston untuk menentukan harga opsi pada model opsi eropa. Model *Heston* merupakan solusi bentuk tertutup untuk opsi *calls* eropa ketika aset berkorelasi dengan *volatility* dan

menyesuaikan model untuk menggabungkan *stochastic interest rate* (Heston, 1993).

Model *Heston* mengasumsikan *underlying stock price* menggunakan proses *stochastic* seperti model *Black-Scholes* (Black & Scholes, 1973) dengan perubahan bahwa *volatility* tidak konstan terhadap waktu atau *stochastic*. Berdasarkan hal tersebut model Heston memiliki dua *stochastic differential equations* (SDEs) (Rouah, 2013).

Model *Heston* membentuk persamaan untuk menghitung nilai opsi model *european*, sebagai berikut:

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t d\mathrm{W}_{1,t} \quad (1)$$

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t} d\mathrm{W}_{2,t} \quad (2)$$

Parameter dari model *Heston* adalah

a. μ merupakan arus proses dari *stock*
b. $\kappa > 0$ merupakan kecepatan pengembalian rata-rata untuk varians
c. $\theta > 0$ merupakan tingkat pengembalian rata-rata untuk varians
d. $\sigma > 0$ merupakan *volatility* varians
e. $v_0 > 0$ merupakan inisial (*time zero*) dari tingkat varians
f. $\rho \in [-1, 1]$ merupakan korelasi antara dua *Brownian motions* W1 dan W2
g. λ merupakan parameter resiko *volatility*

Dalam model *Heston*, *volatility* tidak dimodelkan secara langsung namun berbentuk varians (v_t). Proses varians muncul dari proses *Ornstein-Uhlenbeck* terhadap volatility $h_t = \sqrt{v_t}$ yang diperoleh dari

$$dv_t = (\delta^2 - 2\beta v_t)dt + 2\delta d\sqrt{v_t}W_{2,t} \quad (3)$$

Dengan $\kappa = 2\beta,\ \theta = \delta^2/(2\beta)$ dan $\sigma = 2\delta$ mengekspresikan persamaan dv_t.

Stock price dan varians dalam persamaan dS_t dan dv_t mengikuti proses pengukuran historikal atau pengukuran fisikal. Sedangkan untuk proses valuasi harga (S_t, v_t) memerlukan proses pengukuran risiko netral. Model *Heston* melakukannya dengan memodifikasi persamaan dS_t dan dv_t dengan menerapkan teorema Girsanov. Proses risiko netral untuk *stock price* adalah

$$dS_t = rS_t dt + \sqrt{v_t}S_t dW_{1,t} \quad (4)$$

dimana

$$\widetilde{W}_{1,t} = \left(W_{1,t} + \frac{\mu - r}{\sqrt{v_t}}\right) \quad (5)$$

Jika stock membayarkan dividen yield, q, secara terus menerus maka pada persamaan (4), r digantikan dengan q.

Proses risiko netral untuk varians diperoleh dengan memasukkan *volatility risk premium*, $\lambda(S_t, v_t, t)$, kedalam persamaan dv_t menjadi

$$dv_t = [\kappa(\theta - v_t) - \lambda(S_t, v_t, t)]dt + \sigma\sqrt{v_t}dW_{2,t} \quad (6)$$

dimana

$$\widetilde{W}_{2,t} = \left(W_{2,t} + \frac{\lambda(S_t, v_t, t)}{\sigma\sqrt{v_t}}\right) \quad (7)$$

Heston beranggapan bahwa model konsumsi menghasilkan proporsional premium kepada varians, sehingga $\lambda(S_t, v_t, t) = \lambda v_t$ dimana λ adalah konstan. Berdasarkan itu maka persamaan proses risiko netral untuk varians menjadi

$$dv_t = \kappa^*(\theta^* - v_t)dt + \sigma\sqrt{v_t}d\widetilde{W}_{2,t} \quad (8)$$

Dengan $\kappa^* = \kappa + \lambda$ dan $\theta^* = \kappa\theta/(\kappa + \lambda)$ sebagai parameter risiko netral dari proses varians. Sehingga proses risiko netral untuk stock price dan varians menjadi

$$dS_t = rS_t dt + \sqrt{v_t}S_t d\widetilde{W}_{1,t} \quad (10)$$

$$dv_t = \kappa^*(\theta^* - v_t)dt + \sigma\sqrt{v_t}d\widetilde{W}_{2,t} \quad (11)$$

Parameter $\kappa^* = \kappa$, $\theta^* = \theta$ sama dalam pengukuran fisikal atau risiko netral saat $\lambda = 0$. Sehingga saat melakukan estimasi parameter risiko netral untuk harga opsi tidak perlu melakukan estimasi λ.

3.4 *Finite Difference* untuk Heston PDE

Metode *finite difference* memiliki gagasan untuk mendiskritisasi domain dengan beberapa titik *grids* dan menggunakan *finite difference* untuk memperkirakan derivatif pada titik-titik *grids* ini (Chen, 2017). Metode *Finite Difference* memiliki asumsi bahwa model *grids* dapat berbentuk terstruktur atau tidak terstruktur. Metode *finite difference* menjadi teknik untuk mendapatkan perkiraan numerik dari PDE.

Metode Heston sebagai model stochastic volatility umum, dalam bentuk Heston PDE memiliki argumen yang serupa dengan Black-Scholes PDE. Dalam Black-Scholes PDE, portofolio dibentuk dengan underlying stock dan derivatif tunggal untuk melindungi stock dan membuat portofolio tanpa risiko. Dalam model Heston diperlukan derivatif tunggal untuk melindungi volatility. Portofolio yang terbentuk terdiri dari satu opsi $V = V(S, v, t)$, Δ unit dari saham, dan φ unit opsi lain $U(S, v, t)$ untuk lindung nilai volatilitas. Persamaan portofolio berbentuk

$$\Pi = V + \Delta S + \varphi U \tag{12}$$

Jika portofolio diasumsikan sebagai *self-financing* maka persamaan portofolio menjadi

$$d\Pi = dV + \Delta dS + \varphi dU \tag{13}$$

Bentuk *Heston* PDE untuk opsi dengan *dividend-paying stock*, $\lambda = 0$, *stock price* adalah S, *volatility* adalah v, dan *maturity* adalah t, berbentuk

$$\frac{\partial U}{\partial t} = \frac{1}{2} vS^2 \frac{\partial^2 U}{\partial S^2} + \rho\sigma vS \frac{\partial^2 U}{\partial v \partial S} + \frac{1}{2}\partial^2 v \frac{\partial^2 U}{\partial v^2} - rU + (r-q)S\frac{\partial U}{\partial S} + \kappa(\theta - v)\frac{\partial U}{\partial v} \tag{14}$$

Persamaan tersebut dapat disederhanakan menggunakan $U(t) = U(s, v, t)$ menjadi

$$\frac{\partial U}{\partial t} = LU(t) \tag{15}$$

Dimana L didefinisikan sebagai

$$L = \frac{1}{2} vS^2 \frac{\partial^2}{\partial S^2} + \frac{1}{2}\sigma^2 v \frac{\partial^2}{\partial v^2} + \rho\sigma vS \frac{\partial^2}{\partial v \partial S} + (r-q)S\frac{\partial}{\partial S} + \kappa(\theta - v)\frac{\partial}{\partial v} - r \tag{16}$$

Untuk dapat mengimplementasikan *finite difference* untuk memecahkan Heston PDE, perlu dilakukan diskretisasi *grids* untuk variabel *stock price* dan varians dan diskretisasi *grids* untuk *maturity*. *Grids finite difference* dapat berbentuk *uniform grids* atau *non-uniform grids* dengan karakteristik tersendiri. *Uniform grids* memiliki jarak *grids* yang meningkat secara teratur terhadap dua variabel yang digunakan. Keuntungan *uniform grids* adalah sederhana dan mudah untuk dibangun. *Non-uniform grids* memiliki jarak *grids* yang tidak teratur terhadap dua variabel yang digunakan. Hal tersebut menyebakan *non-uniform grids* memiliki konstruksi dan perhitungan yang rumit. Disamping kerumitannya, *non-uniform grids* dapat diperhalus di titik-titik tertentu sehingga untuk valuasi harga opsi dapat dihasilkan harga yang akurat dengan menggunakan sedikit titik *grids*.

Variabel yang digunakan untuk membentuk grids adalah S, v, dan t. Perlu ditentukan nilai maksimal dan nilai minimum dari S, v, dan t sebagai batas nilai. Nilai maksimal dinotasikan sebagai S_{max}, v_{max}, dan t_{max}. Nilai S_{max} dan v_{max} didapatkan berdasarkan kasus opsi yang diperhitungkan, sedangkan $t_{max} = \tau$ berdasarkan waktu *maturity*. Nilai minimum dinotasikan sebagai S_{min}, v_{min}, dan t_{min}. Nilai minimum akan selalu diatur bernilai $S_{min} = v_{min} = t_{min} = 0$ sebagai batas bawah.

Ukuran *grids* diatur dengan $N_s + 1$ titik untuk *stock price*, $N_v + 1$ titik untuk *volatility*, dan $N_T + 1$ titik untuk *maturity*. Lebar *non-uniform grids* untuk $N_s + 1$ *stock price* diatur dengan persamaan

$$S_i = K + c\sinh(\xi_i), i = 0,1, \dots, N_s \quad (17)$$

dimana $c = K/5$ dan

$$\xi_i = sinh^{-1}(-K/c) + i\Delta\xi \quad (18)$$

dengan

$$\Delta\xi = \frac{1}{N_S}\left[sinh^{-1}\left(\frac{S_{max}-K}{c}\right) - sinh^{-1}\left(\frac{K}{c}\right)\right] \quad (19)$$

Lebar *non-uniform grids* untuk $N_v + 1$ *volatility* diatur dengan persamaan

$$v_i = d\sinh(j\Delta\eta), j = 0,1,\dots,N_V \quad (20)$$

dengan

$$\Delta\eta = \frac{1}{N_V} sinh^{-1}\left(\frac{V_{max}}{d}\right) \quad (21)$$

dimana $d = V_{max}/500$.

Lebar non-uniform grids untuk $N_t + 1$ volatility diatur dengan persamaan

$$t_i = i \times dt, n = 0,1,\dots,N_T \quad (22)$$

dengan

$$d_t = (t_{max} - t_{min})/N_T \quad (23)$$

Non-uniform grids yang terbentuk akan lebih halus di sekitar strike price K dan di sekitar titik *volatility* $v_0 = 0$.

Model *Heston* PDE ini memperkirakan nilai titik di bagian interior dan bagian batas secara terpisah. Bagian interior (S_i, v_i) dilakukan perkiraan nilai

menggunakan *first-order* derivatif dengan beda tengah (*central difference*) sebagai berikut

$$\frac{\partial U}{\partial S}(S_i, v_i) = \frac{U^n_{i+1,j} - U^n_{i-1,j}}{S_{i+1} - S_{i-1}},\ \frac{\partial U}{\partial v}(S_i, v_i) = \frac{U^n_{i,j+1} - U^n_{i,j-1}}{v_{j+1} - v_{j-1}} \quad (24)$$

Bagian batas memiliki beberapa kondisi yang perlu di inisialisasi, yaitu kondisi saat *maturity*, $\mathrm{S} = S_{min}$ $\mathrm{S} = S_{max}$, $\mathrm{V} = V_{max}$, dan $\mathrm{V} = V_{min}$.

Kondisi batas saat *maturity*, $\mathrm{t} = 0$, nilai dari opsi *calls* adalah nilai instrinsiknya (*payoff*) sehingga didapat persamaan

$$U(S_i, v_i, 0) = max(0, S_i - K) \quad (25)$$

dengan batas $i = 0, 1, \dots, N_s$ dan $j = 0, 1, \dots, N_v$. Berdasarkan persamaan tersebut maka nilai vektor U^0 adalah nol.

Kondisi batas saat $\mathrm{S} = S_{min} = 0$, opsi *calls* menjadi tidak berguna. Karena itu diperoleh $\mathrm{U}(0, v_i, t_n) = 0$ sehingga kondisi batas menjadi $U^n_{0,j} = 0$ dengan batas $n = 0, 1, \dots, N_T$ dan $j = 0, 1, \dots, N_v$.

Kondisi batas saat $\mathrm{S} = S_{max}$, persamaan yang digunakan adalah $\frac{\partial U}{\partial S}(S_{max}, v_i, t_n) = 1$ sehingga kondisi batas menjadi $U^n_{N_s,j} = S_{max}$ dengan batas $n = 0, 1, \dots, N_T$ dan $j = 0, 1, \dots, N_v$.

Kondisi batas saat $\mathrm{V} = V_{max}$, menggunakan kondisi batas $U^n_{i,N_v} = S$ dengan batas $n = 0, 1, \dots, N_T$ dan $i = 0, 1, \dots, N_s$. Dalam kondisi V_{max} diperoleh $U = S$ dan $\frac{\partial U}{\partial S} = 1$.

Kondisi batas saat $\mathrm{V} = V_{min} = 0$, persamaan yang digunakan ada 2 yaitu $\frac{\partial U}{\partial S}$ yang diselesaikan menggunakan beda tengah (*central difference*) dan $\frac{\partial U}{\partial v}$ yang diselesaikan menggunakan beda maju (*forward difference*). Persamaan yang terbentuk adalah

$$\frac{\partial U}{\partial S}(S_i, 0, t_n) = \frac{U^n_{i+1,0} - U^n_{i-1,0}}{S_{i+1} - S_{i-1}},\ \frac{\partial U}{\partial v}(S_i, 0, t_n) = \frac{U^n_{i,1} - U^n_{i,0}}{v_1} \tag{26}$$

Skema *explicit* akan digunakan sebagai teknik menyelesaikan *finite difference*. Persamaan yang digunakan untuk memperoleh elemen $U^{n+1}_{i,\mathrm{j}}$ adalah

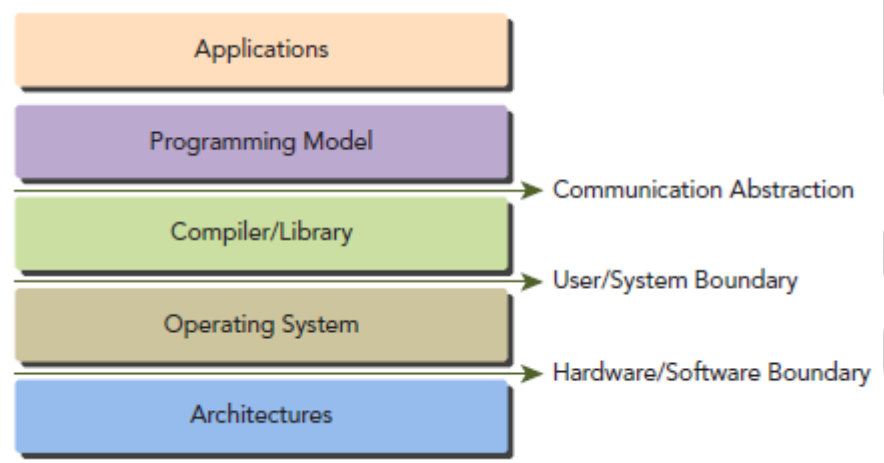
$$U^{n+1}_{i,j} = U^n_{i,j} + dt\left[\frac{1}{2}v_j S_i^2 \frac{\partial^2}{\partial S^2} + \frac{1}{2}\sigma^2 v_j \frac{\partial^2}{\partial V^2} + \rho\sigma v_j S_i \frac{\partial^2}{\partial v \partial S} + (r-q)S_i \frac{\partial}{\partial S} + \kappa(\theta - v_j)\frac{\partial}{\partial v} - r\right] U^n_{i,j} \tag{27}$$

3.5 Pemrograman Paralel

Pemrograman paralel merupakan algoritma pemrograman yang membentuk program yang mampu mengerjakan beberapa proses secara paralel memanfaatkan *multiple processor* atau *multiple computer*. Pengerjaan proses secara paralel dilakukan untuk mengatasi permasalahan pemrosesan yang tidak dapat diselesaikan menggunakan *single processor* atau memerlukan waktu yang lama. Pemrograman paralel akan dapat menyelesaikan pekerjaan yang berskala besar dan rumit secara berasamaan dengan *throughput* dan efisiensi tinggi.

3.5.1 Compute Unified Device Architecture (CUDA) Programming Model

CUDA merupakan platform pemrograman paralel menggunakan GPU yang berbasis bahasa pemrograman C. Pemrograman CUDA dapat dilakukan di berbagai tools populer yang berbasis bahasa pemrograman C, untuk melakukan kegiatan edit, debug dan analyze program. CUDA Programming Model menjadi jembatan komunikasi antara aplikasi dengan perangkat keras. Komunikasi yang dikirimkan menentukan bagaimana komponen program berbagi informasi dan mengoordinasikan kegiatan mereka (Cheng, Grossman, & McKercher, 2014). Layers antara program dengan implementasi model pemrogramannya dapat dilihat pada Gambar 3.1.



Gambar 3.1. CUDA Programming Model

Architectures pada model pemrograman CUDA memiliki fitur-fitur yang dapat dimanfaatkan dalam komputasi GPU antara lain mengatur dan mengakses memori pada GPU melalui struktur hirarki. Penggunaan fitur-fitur tersebut membutuhkan operating system dan hardware sebagai tempat user menulis program dengan menggunakan compiler/library untuk menentukan bagaimana kompenen

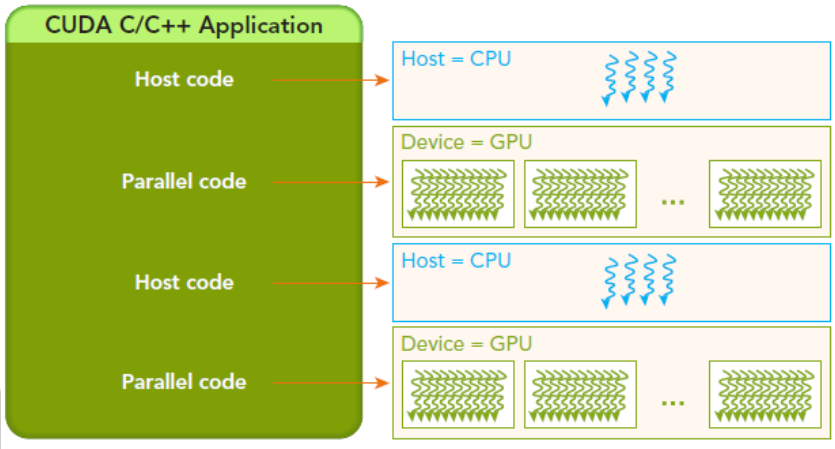
program berbagi informasi dan mengoordinasikan kegiatan mereka. Selain itu, communication abstraction merupakan batas antara program dan implementasi programming model dalam aplikasinya menyediakan logical view dari arsitektur komputasi yang spesifik berupa komputasi paralel dari berbagai level, seperti tingkat domain, logika dan perangkat keras.

3.5.2 Compute Unified Device Architecture (CUDA) Programming Structure

Model pemrograman CUDA dapat mengksekusi aplikasi pada sistem komputasi heterogen dengan hanya menganotasikan kode dengan seperangkat ekstensi ke bahasa pemrograman C. Lingkungan heterogen berupa CPU dengan memorinya yang dapat disebut sebagai host dan dilengkapi GPU dengan memorinya yang dapat disebut sebagai device dan dipisahkan oleh PCI-Express bus.

NVDIA memperkenalkan Unified Memory yang merupakan perbaikan model pemrograman CUDA dan berfungsi untuk menjembatani pembagian antara memori host dan device dengan menggunakan single pointer. Penggunaan memori host dan device memerlukan pengalokasian yang tepat sehingga dapat mengoptimalkan aplikasi yang dibuat dan memaksimalkan penggunaan perangkat keras.

Gambar 3.2 menjelaskan struktur proses pengaplikasian CUDA yang terdiri dari kode serial yang dilengkapi kode paralel.



Gambar 3.2. CUDA Programming Structure

Kode serial dijalankan pada host, sementara kode paralel dijalankan pada device GPU. Host code ditulis dalam ANSI C dan Device code ditulis menggunakan CUDA C. Semua kode dapat diletakkan dalam single source file atau dapat menggunakan beberapa sumber file untuk membangun aplikasi atau pustaka yang diinginkan. Kode yang telah dibuat untuk host dan device dapat dijalankan menggunakan NVIDIA C Compiler (nvcc). Selain itu, alur pemrosesan program CUDA sebagai berikut:

- Salin data dari memori CPU ke memori GPU
- Aktifkan kernel untuk beroperasi pada data yang disimpan dalam memori GPU
- Salin kembali data dari memori GPU ke memori CPU