OPEN ACCESS
Indo-JC Indonesia Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 5, Issue. 1, March 2020. pp. 93-102
doi:10.21108/indojc.2020.5.1.395

Performansi Implementasi Paralel OpenMP pada Persamaan Air Dangkal 2D untuk Simulasi Gelombang Runup

Novalianda Jeriano[1], Didit Adytia [*1]

[1]*School of Computing, Telkom University*

Jl. Telekomunikasi 1, Bandung, 40257, Indonesia

[*] adytia@telkomuniversity.ac.id

Abstract

Many shallow flows can be found in nature such as tsunami, flooding in coastal cities, flows in river channel, tidal, etc. These shallow flows can be simulated by using the well-known Shallow Water Equations (SWE). One important nonlinear phenomenon in the shallow flows is the runup phenomenon. Especially for simulating accurately runup phenomenon, special treatment in numerical implementation of the model should be considered. In this paper, we implemented the 2D SWE with the Finite Volume method on momentum conservative *staggered grid* for numerical implementation. To improve computational performance especially for simulating large computational domain with high resolution grid, we parallelize the numerical scheme by using OpenMP architecture. Performance of the parallel implementation is measured by calculating the speedup and efficiency. From the results of the parallelization, the maximum speedup is 3.95 times for 4 threads and 6.61 times for 8 threads. The maximum efficiency in computational time is obtained 91.4% for 4 threads and 82.7% for 8 threads for cases with a large number of computing grids.

Keywords: OpenMP, parallel, Shallow Water Equations, *staggered grid*, Shallow Flows

Abstrak

Terdapat banyak aliran dangkal di alam seperti tsunami, aliran banjir, aliran pada sungai, pasang surut air laut, dan sebagainya. Aliran dangkal dapat disimulasikan dengan menggunakan Persamaan Air Dangkal atau Shallow Water Equations (SWE). Salah satu fenomena nonlinear yang penting dalam aliran dangkal adalah fenomena runup. Khususnya untuk menyimulasikan fenomena runup secara akurat, perlakuan khusus pada implementasi numerik dari model gelombang harus dilakukan. Pada artikel ini, persamaan SWE diimplementasikan dengan metode Finite Volume pada grid komputasi dengan model momentum conservative *staggered grid*. Untuk meningkatkan performasi komputasi terutama untuk menyimulasikan domain komputasi yang besar dengan resolusi grid tinggi, pada paper ini skema numerik tersebut diimplementasikan dengan metode arsitektur OpenMP. Performansi algoritma paralel dikuantifikasi dengan menghitung *speedup* dan efisiensi. Dari hasil paralelisasi tersebut, didapatkan *speedup* maksimum mencapai 3.95 kali untuk 4 thread dan 6.61 kali untuk 8 thread. Sedangkan efisiensi maksimum didapatkan pada waktu komputasi adalah 91.4% untuk 4 thread dan 82.7% untuk 8 thread untuk kasus-kasus dengan jumlah grid komputasi yang besar.

Kata Kunci: OpenMP, Paralel, Persamaan Air Dangkal, *staggered grid*, Aliran Dangkal

Received on Jan 2020. Accepted on March 2020

I. Pendahuluan

ALIRAN air dangkal dapat ditemukan pada berbagai fenomena alam, seperti propagasi gelombang tsunami, aliran pasang surut di laut, aliran air sungai, aliran banjir pada perkotaan, dan sebagainya. Aliran air dangkal ini sangat penting untuk dipelajari, misalnya untuk prediksi waktu datangnya gelombang tsunami, prediksi daerah rendaman tsunami, prediksi rendaman banjir untuk perencanaan pembangunan pada daerah pesisir dan pada daerah-daerah yang berpotensi banjir, simulasi dan prediksi potensi banjir pada perkotaan, simulasi longsor bawah laut [1].

Fenomena aliran air dangkal dapat dimodelkan dengan menggunakan Persamaan Air Dangkal atau yang dikenal dengan nama *Shallow Water Equations* (SWE) atau persamaan Saint-Venant. Persamaan ini terdiri dari persamaan kontinuitas dan persamaan momentum, yang merepresentasikan konservasi massa dan momentum [2]. Secara umum, untuk memecahkan persamaan SWE secara numerik dapat dilakukan dengan metode numerik seperti *Finite Volume Method*, *Finite Element Method* dan *Finite Difference Method* (lihat [3], [5], [10]). Akurasi dan efisiensi dari ketiga metode numerik ini bervariasi, dimana pada umumnya, untuk melakukan simulasi dengan jumlah grid komputasi yang besar, ketiga metode numerik tersebut membutuhkan beban komputasi yang besar.

Salah satu cara untuk mereduksi waktu komputasi adalah dengan menerapkan algoritma paralel pada implementasi numerik yang dipilih. Pemrograman paralel adalah teknik untuk memungkinkan pelaksanaan operasi simulasi menggunakan lebih dari satu unit pemrosesan. Teknik ini dapat membuat waktu eksekusi program lebih cepat dari eksekusi serial biasa. Ada banyak arsitektur untuk memparalelkan model ini, seperti Central Processing Unit (CPU) dan Graphics Processing Unit (GPU). Pemrograman paralel menggunakan CPU dapat menggunakan arsitektur *Shared Memory* (OpenMP) dan *Message Processing Interface* (MPI). OpenMP adalah Antarmuka Pemrograman Aplikasi (API) yang menyediakan multithreading dengan shared memory. Multithreading adalah kemampuan CPU untuk menjalankan beberapa proses pada saat yang bersamaan. Ini dapat mengurangi proses runtime dan meningkatkan efisiensi suatu algoritma. Selain itu, shared memory adalah arsitektur paralel yang memungkinkan setiap proses yang berjalan untuk mendapatkan sumber memori secara proporsional. Selain itu, waktu akses ke penyimpanan dapat minimum dan mempercepat proses runtime dapat ditingkatkan [15]. Dalam model OpenMP, komunikasi antara thread diperoleh dengan membaca dan menulis langsung ke shared memory. Dengan menggunakan OpenMP, mudah untuk memecahkan masalah pemrograman seperti load balancing karena OpenMP melakukannya secara otomatis dan tidak perlu mempertimbangkan bagian perhitungan mana yang harus dilakukan pada core [14]. Penggunaan CPU untuk pemrograman paralel membuat *running time* menjadi lebih cepat dan memungkinkan untuk meringankan beban komputasinya.

Performansi dari arsitektur paralel akan diukur menggunakan dua macam pengukuran, yaitu *speedup* dan efisiensi. *Speedup* dari arsitektur OpenMP dicari untuk melihat seberapa cepat performa pemrograman paralel dibanding serial, sedangkan efisiensi dicari untuk melihat ukuran performansi yang dilihat dari segi beban. Penelitian berbasis paralel OpenMP juga telah dilakukan, misalnya Zhang S 2014, [14] meneliti model dam-break flow dengan metode *Finite Volume* menggunakan OpenMP pada komputer multi-core dan juga P.H. Gunawan 2019, [16] diteliti Arsitektur OpenMP dan MPI untuk menyimulasikan osilasi air 1D pada parabola. Pada artikel ini, platform OpenMP arsitektur paralel akan digunakan untuk menyimulasikan persamaan air dangkal dua dimensi (SWE2D) dengan menggunakan metode *Finite Volume* dengan skema *staggered grid* (lihat [4], [5], [7], [10]). Skema *staggered grid* digunakan untuk menyimulasikan fenomena *runup*, yang sangat berguna untuk menyimulasi *runup* gelombang tsunami, simulasi banjir pada kota, dsb. Performansi dari implementasi arsitektur paralel pada skema numerik ini akan dikuantifikasi dengan *speedup* dan efisiensi. Lebih jauh, *speedup* akan diukur dengan membandingkan waktu eksekusi dari paralel dan serial untuk kasus *runup* yang diusulkan.

Struktur penulisan pada artikel ini adalah sebagai berikut. Pada bagian selanjutnya akan dijelaskan tentang persamaan SWE 2D dan implementasi numeriknya dengan menggunakan metode *Finite Volume* dengan *skema staggered grid*. Setelah itu, performansi pemrograman paralel menggunakan OpenMP dan pengukurannya akan dideskripsikan pada bagian 3. Selanjutnya, hasil dan diskusi kasus beserta performansi paralel akan disajikan di bagian 4. Pada bagian terakhir, kesimpulan dari penelitian ini akan dijelaskan pada bagian 5.

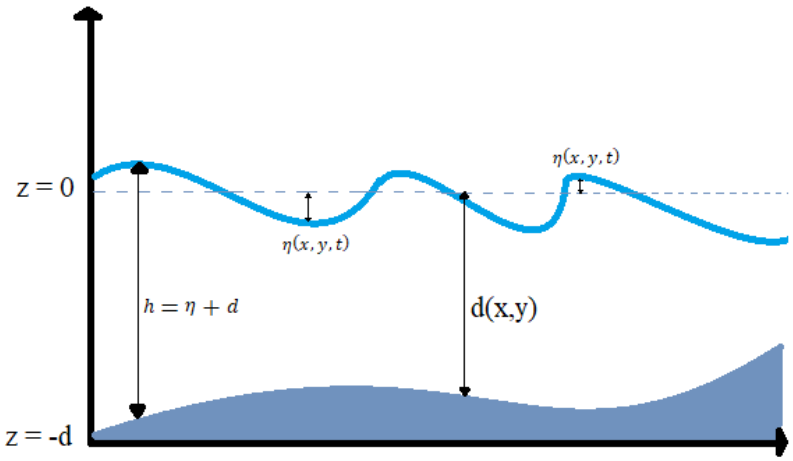
II. MODEL GELOMBANG

A. Shallow Water Equations

Shallow Water Equation (SWE) adalah sistem persamaan diferensial parsial non-linear yang terdiri dari dua persamaan, yaitu persamaan kontinuitas yang mewakili hukum konservasi massa dan persamaan momentum yang mewakili konservasi momentum. Persamaan air dangkal untuk dua dimensi (2D) diberikan sebagai berikut

$$\eta_t + (hu)_x + (hv)_y = 0 \quad (1)$$
$$u_t + uu_x + vu_y + g\eta_x = 0 \quad (2)$$
$$v_t + uv_x + vv_y + g\eta_y = 0 \quad (3)$$

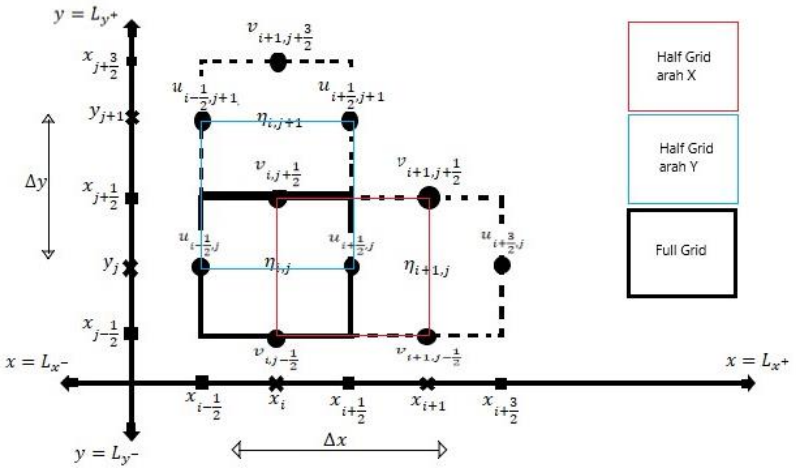
di mana *t* adalah variabel waktu, *x* dan *y* variabel spasial berdasarkan koordinat Cartesius, η *(x, y, t)* menunjukkan ketinggian permukaan air dari keadaan keseimbangan pada posisi *(x, y)* dan waktu *t*, *u (x, y, t)* kecepatan horizontal arah *x* pada posisi *(x, y)* dan waktu *t*, *v (x, y, t)* kecepatan vertikal arah y pada posisi *(x, y)* dan waktu *t*, *d (x,y)* kedalaman air terhadap sumbu *z = 0*, *h (x, y, t)* total kedalaman air ke permukaan pada posisi *(x, y)* dan waktu *t*, dan *g* percepatan gravitasi. Gambar 1 mengilustrasikan variabel-variabel pada persamaan SWE.



Gambar 1. Illustrasi variabel-variabel pada persamaan SWE.

B. Implementasi Numerik

Dalam skema *staggered grid*, titik diskrit dari variabel yang tidak diketahui didiskritkan dalam posisi *staggered*. Pada SWE 2D diberikan domain spasial $\Omega = [L_{x-}, L_{x+}] \times [L_{y-}, L_{y+}]$ dan interval waktu $[0, T]$. Kontrol volume pada konservasi massa disebut *full grid* yang didefinisikan pada $\left(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right) \times \left(y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}\right)$. Sedangkan kontrol volume pada konservasi momentum disebut *half grid*. Pada *half grid* terdapat arah horizontal sumbu x yang ditentukan dengan $(x_i, x_{i+1}) \times \left(y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}\right)$ dan untuk arah vertikal sumbu y ditentukan dengan $\left(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right) \times (y_j, y_{j+1})$. Gambar 2 menunjukkan kontrol volume pada *staggered grid* untuk persamaan air dangkal dua dimensi dengan titik diskrit i dan j. Ada variable dimana η didefinisikan pada (x_i, y_j), u didefinisikan pada $\left(x_{i+\frac{1}{2}}, y_j\right)$ dan v didefinisikan pada $\left(x_i, y_{j+\frac{1}{2}}\right)$.



Gambar 2. Kontrol volume pada skema *staggered grid*.

Persamaan (1) diapproksimasi pada (x_i, y_j) sebagai berikut

$$\eta_{i,j}^{n+1} - \eta_{i,j}^{n} = -\frac{\Delta t}{\Delta x}\left((*_{hu})|_{i+\frac{1}{2},j}^{n} - (*_{hu})|_{i-\frac{1}{2},j}^{n}\right) - \frac{\Delta t}{\Delta y}\left((*_{hv})|_{i,j+\frac{1}{2}}^{n} - (*_{hv})|_{i,j-\frac{1}{2}}^{n}\right) \tag{4}$$

Pada sel massa, nilai yang menunjukkan $*_h$ hilang pada titik half grid, sehingga nilai $*_h$ dihampiri oleh first-order upwinding sebagai berikut

$${}^{*}h_{i+\frac{1}{2},j}^{n} = \begin{cases} h_{i,j}^{n} & if\ u_{i+\frac{1}{2},j}^{n} \geq 0 \\ h_{i+1,j}^{n} & if\ u_{i+\frac{1}{2},j}^{n} < 0 \end{cases} \tag{5}$$

$${}^{*}h_{i,j+\frac{1}{2}}^{n} = \begin{cases} h_{i,j}^{n} & if\ v_{i+\frac{1}{2},j}^{n} \geq 0 \\ h_{i,j+1}^{n} & if\ v_{i+\frac{1}{2},j}^{n} < 0 \end{cases} \tag{6}$$

Diskritisasi spasial pada konservasi momentum untuk persamaan air dangkal (2 - 3) yang sudah diaprokksimasi pada titik $\left(x_{i+\frac{1}{2}}, y_j\right)$ dan $\left(x_i, y_{j+\frac{1}{2}}\right)$ diberikan sebagai berikut

$$\begin{aligned}(uu_x)_{i+\frac{1}{2},j}^{n} &= \frac{1}{\bar{h}_{i+\frac{1}{2},j}^{n}}\left(\frac{{}_{u}\bar{q}_{i+1,j}^{n}\ {}^{*}u_{i+1,j}^{n} - {}_{u}\bar{q}_{i,j}^{n}\ {}^{*}u_{i,j}^{n}}{\Delta x} - u_{i+\frac{1}{2},j}^{n}\frac{{}_{u}\bar{q}_{i+1,j}^{n} - {}_{u}\bar{q}_{i,j}^{n}}{\Delta x}\right) \\ (vu_y)_{i+\frac{1}{2},j}^{n} &= \frac{1}{\bar{h}_{i+\frac{1}{2},j}^{n}}\left(\frac{{}_{v}\bar{q}_{i+1,j}^{n}\ {}^{*}u_{i+1,j}^{n} - {}_{v}\bar{q}_{i,j}^{n}\ {}^{*}u_{i,j}^{n}}{\Delta y} - u_{i+\frac{1}{2},j}^{n}\frac{{}_{v}\bar{q}_{i+1,j}^{n} - {}_{v}\bar{q}_{i,j}^{n}}{\Delta y}\right)\end{aligned} \tag{7}$$

$$(uv_x)_{i,j+\frac{1}{2}}^{n} = \frac{1}{\bar{h}_{i,j+\frac{1}{2}}^{n}}\left(\frac{{}_{u}\bar{q}_{i,j+1}^{n}\ {}^{*}v_{i,j+1}^{n} - {}_{u}\bar{q}_{i,j}^{n}\ {}^{*}v_{i,j}^{n}}{\Delta x} - v_{i,j+\frac{1}{2}}^{n}\frac{{}_{u}\bar{q}_{i,j+1}^{n} - {}_{u}\bar{q}_{i,j}^{n}}{\Delta x}\right) \tag{8}$$

$$(vv_y)^n_{i,j+\frac{1}{2}} = \frac{1}{\bar{h}^n_{i,j+\frac{1}{2}}}\left(\frac{{}_v\bar{q}^n_{i,j+1}\ {}^*v^n_{i,j+1} - {}_v\bar{q}^n_{i,j}\ {}^*v^n_{i,j}}{\Delta y} - u^n_{i,j+\frac{1}{2}}\frac{{}_v\bar{q}^n_{i,j+1} - {}_v\bar{q}^n_{i,j}}{\Delta y}\right)$$

Dimana ${}_uq = hu$ dan ${}_vq = hv$, masing-masing merupakan momentum arah x dan y, dan nilai lain seperti u dan v didefinisikan pada first-order upwinding sebagai berikut

$${}^*u^n_{i,j} = \begin{cases} u^n_{i-\frac{1}{2},j} & if\ {}_u\bar{q}^n_{i,j} \geq 0 \\ u^n_{i+\frac{1}{2},j} & if\ {}_u\bar{q}^n_{i,j} < 0 \end{cases} \quad (9)$$

$${}^*v^n_{i,j} = \begin{cases} v^n_{i,j-\frac{1}{2}} & if\ {}_v\bar{q}^n_{i,j} \geq 0 \\ u^n_{i,j+\frac{1}{2}} & if\ {}_v\bar{q}^n_{i,j} < 0 \end{cases} \quad (10)$$

III. IMPLEMENTASI PARALEL OPENMP

OpenMP adalah antarmuka pemrograman aplikasi yang menyediakan multi-pemrosesan memori bersama dalam pemrograman C / C ++. Implementasi OpenMP relatif sederhana dan mudah untuk diterapkan terutama pada algoritma yang telah di program secara efisien. Pada arsitektur openMP, setiap thread memungkinkan untuk mengakses memori utama untuk berkomunikasi data. Pada bahasa pemrograman C / C ++, bagian-bagian yang memungkinkan untuk diparalelisasi cukup hanya dengan menambahkan sintaks "#pragma omp parallel for" di atas kode perulangan. Selain itu, sintaks "collapse (2)" dapat ditambahkan jika memiliki perulangan ganda. Secara umum, setiap thread akan mengeksekusi kode secara paralel dan independen, sehingga hal ini akan meminimalkan waktu akses ke penyimpanan dan dapat mempercepat waktu pemrosesan. Proses blok pemrograman paralel menggunakan OpenMP untuk persamaan air dangkal dalam 2D dengan skema *staggered grid* dapat dilihat pada Gambar 3.

Dari Gambar 3 ditunjukkan bahwa blok serial dan paralel memiliki perbedaan proses, dimulai dari pendeklarasian variabel dan parameter yang dilakukan pada blok serial. Lalu, dilanjutkan dengan pembuatan kondisi awal dimana terdapat grid x_i dan y_j, kedalaman $(d_{i,j})$, elevasi permukaan $(\eta_{i,j})$, total kedalaman $(h_{i,j})$ dan kecepatan horizontal & vertikal. Selanjutnya, menentukan nilai $*_h$ menggunakan skema upwind untuk menghitung flux ${}_uq$ dan ${}_vq$. Setelah itu, menghitung persamaan kontinuitas dan momentum hingga mendapatkan nilai baru dari $(\eta_{i,j})$ dan kecepatan. Perhitungan pada blok paralel akan dilakukan selama waktu final.

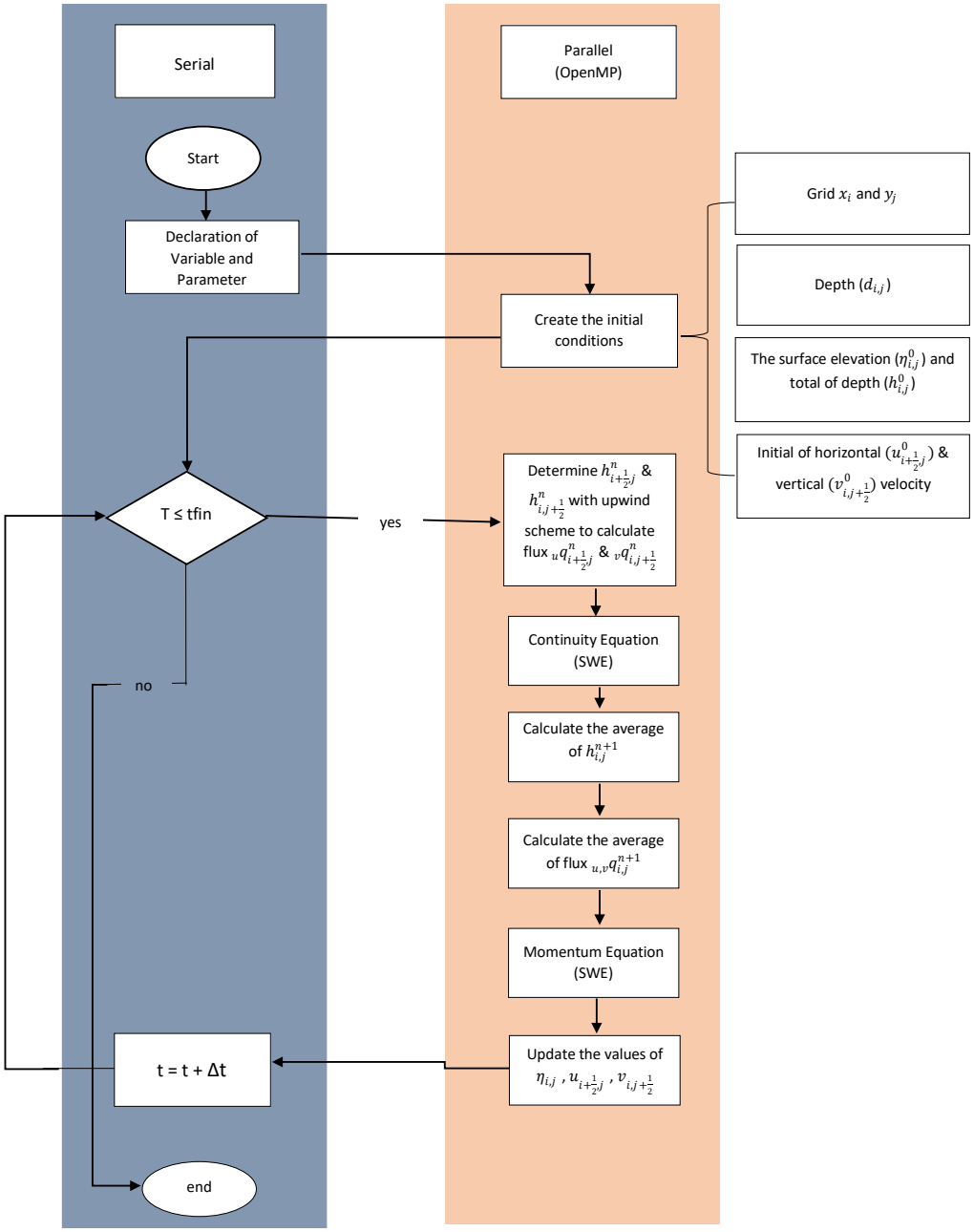
Untuk mencari performa arsitektur paralel dapat dilihat dengan dua pengukuran, yaitu *speedup* dan efisiensi. *Speedup* dapat diperoleh dengan

$$S(p) = \frac{T_s}{T_p} \quad (11)$$

dimana $S(p)$ adalah *speedup*, T_s waktu CPU pemrosesan serial dan T_p waktu CPU pemrosesan paralel. Sedangkan, Efisiensi dapat diperoleh dengan

$$E_{(p)} = \frac{S(p)}{N} \times 100\% \quad (12)$$

dimana $E_{(p)}$ adalah efisiensi, N adalah jumlah thread yang digunakan dalam pemrosesan paralel.



Gambar 3. Proses blok pemrograman paralel menggunakan OpenMP

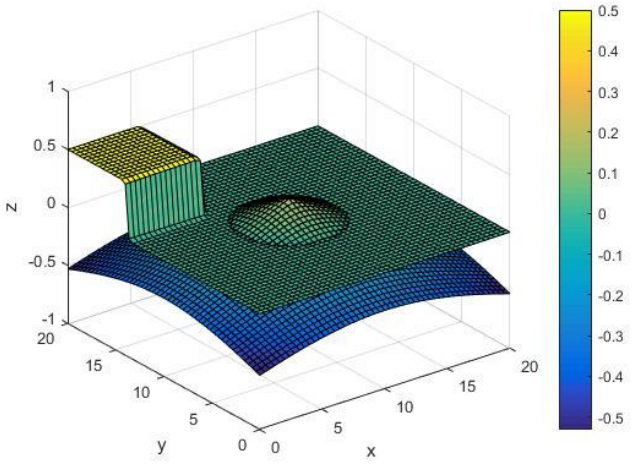
IV. Hasil dan Diskusi

Pada bagian ini, simulasi numerik dengan kasus pulau *conical* ditampilkan pada Gambar 4. Kondisi awal simulasi untuk elevasi gelombang $\eta(x, y, 0)$ diberikan sebagai berikut

$$\eta(x, y, 0) = \begin{cases} \max(A0, -d(x,y)) & if \quad x \leq 6, y \geq 14 \\ \max(0, -d(x,y)) & if \quad x > 6, y < 14 \end{cases} \tag{13}$$

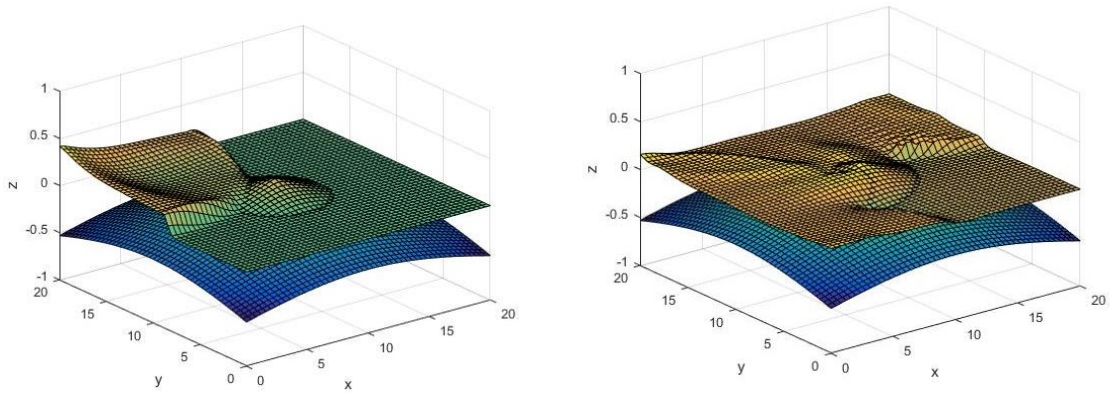
Sedangkan bentuk topografi pulau *conical* untuk merepresentasikan gelombang runup diberikan oleh rumus berikut

$$d(x,y) = -1 - (r\alpha - H_0 - \sqrt{(\alpha^2[(x-x_0)^2 + (y-y_0)^2])} \quad (14)$$



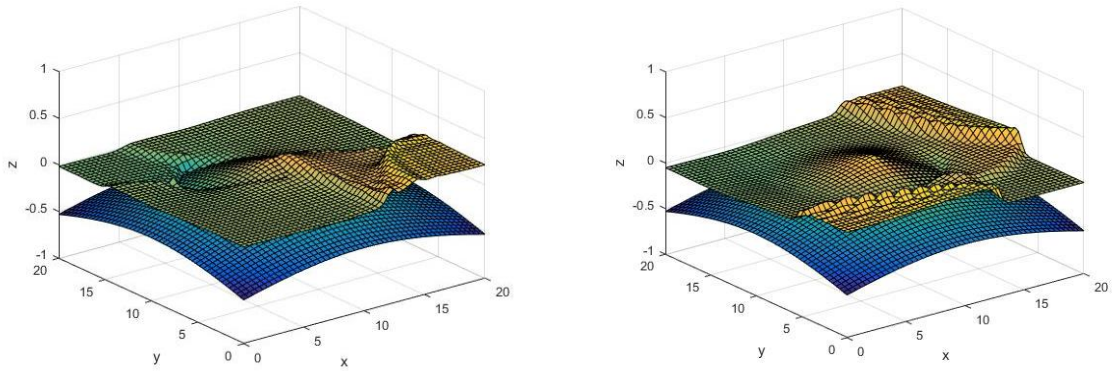
Gambar 4. Kondisi awal elevasi gelombang dan topografi yang digunakan.

Dimana paramater r = 3.6 adalah jarak dari titik pusat, $\alpha = \frac{1}{19.85}$ adalah kemiringan pulau, H_0 = 1 adalah kedalaman air pada area datar dan $A0$ = 0.5 adalah amplitudo. Pada Gambar 4 menunjukkan kondisi awal, dengan ketinggian air digambarkan oleh area kuning dan bentuk pulau digambarkan oleh area biru. Sehingga,



simulasinya akan menggambarkan sebuah pulau yang akan terkena gelombang runup pada Gambar 5.

t = 1.998 s t = 7.9824 s



Gambar 5. Simulasi pada t=2, t=8, t=13 dan t=25

Untuk menyimulasikan kasus ini, kinerja pemrosesan paralel akan disajikan. Spesifikasi komputer yang akan digunakan ditunjukkan pada Tabel. I

TABEL I
SPESIFIKASI KOMPUTER

Name	Type
Operating System	Centos 7.5
Processors	Intel(R) Xeon(R) CPU E5-2670 @2.3GHz
RAM	8 GB

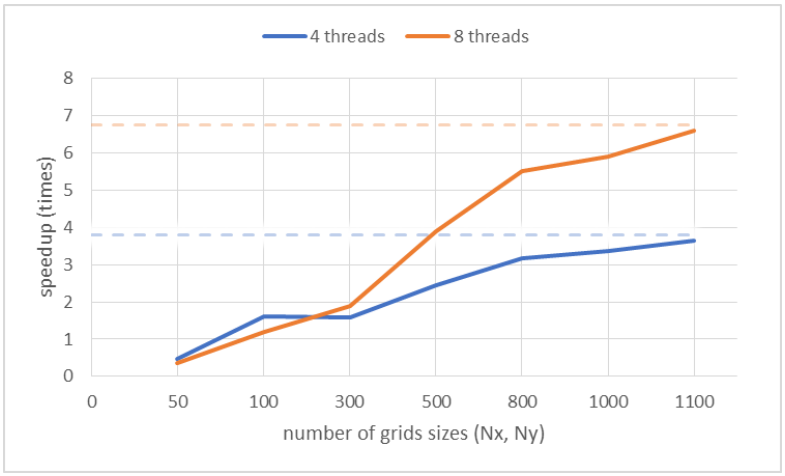
Performansi ini akan dilakukan dengan memeriksa waktu CPU untuk berbagai nomor grid dari ukuran 51x51 sampai 1101x1101. Waktu CPU untuk pemrosesan serial dan paralel menggunakan OpenMP dengan 4 dan 8 thread di waktu terakhir 25 ditampilkan di Tabel. II

TABEL II
WAKTU CPU PADA SERIAL DAN PARALEL

Jumlah Grid	Serial (s)	Parallel OpenMP (s)	
		4 Threads	*8 Threads*
51 x 51	0.9277	2.0512	2.6134
301 x 301	54.4988	34.4402	32.0230
801 x 801	329.2133	103.3529	61.6183
1001 x 1001	402.7949	119.7069	68.2822
1101 x 1101	483.6001	132.6801	73.1951

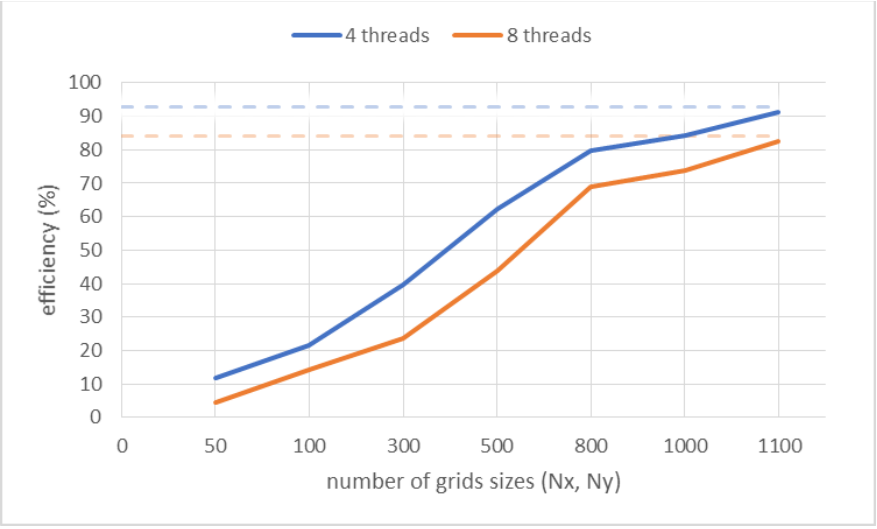
Waktu CPU pada pemrosesan serial dan paralel dilakukan dengan lima kali percobaan pada jumlah grid yang berbeda, untuk jumlah grid (51, 51) didapatkan waktu CPU serial selama 1.133s, 1.114s, 0.742s, 0.964s, 0.899s. Pada waktu CPU paralel dengan 4 threads didapatkan waktu selama 2.138s, 2.004s, 2.142s, 2.022s, 1.901s dan 8 threads selama 2.544s, 2.531s, 2.638s, 2.691, 2.613s, sehingga pada Tabel II menunjukkan rata-rata waktu CPU dari berbagai grid. Dari Tabel II, waktu CPU untuk pemrosesan serial pada jumlah grid (51, 51) lebih baik daripada pemrosesan paralel. Itu karena arsitektur paralel memerlukan waktu untuk berkomunikasi antara prosesor, sehingga waktu komputasi lebih lama. Namun, untuk sejumlah grid lain (dari (301, 301) hingga (1101, 1101)) lebih baik untuk pemrosesan paralel.

Selanjutnya, eksekusi program yang sudah diparalelkan akan diukur kecepatannya untuk mengetahui seberapa cepat dari eksekusi serialnya. Untuk mengukur performansi *speedup* dan efisiensi dapat dicari menggunakan rumus (11) & (12). Grafik *speedup* akan ditunjukkan pada Gambar 6



Gambar 6. *Speedup* dari OpenMP dengan berbagai jumlah grid

Pada Gambar 6, performansi paralel menggunakan 4 thread memiliki kecepatan sekitar 3.95 kali lebih cepat dan untuk 8 thread sekitar 6.61 kali pada jumlah grid (1101, 1101). Sehingga, untuk mencapai *speedup* maksimum itu akan membutuhkan lebih banyak thread. Efisiensi digunakan untuk mengukur arsitektur paralel yang dilihat dari segi beban. Untuk pengukuran efisiensi akan diberikan dalam bentuk grafik pada Gambar 7.



Gambar 7. Efisiensi dari OpenMP dengan berbagai jumlah grid

Gambar 7 menunjukkan perbandingan efisiensi antara 2 thread berbeda (4 dan 8 thread). Efisiensi menggunakan 4 thread mencapai sekitar 91.4% dan 8 thread mencapai sekitar 82.7%. Sehingga, untuk mendapatkan efisiensi yang lebih baik tidak selalu tergantung pada banyak penggunaan thread.

V. Kesimpulan

Pada artikel ini telah diimplementasikan arsitektur paralel OpenMP untuk model gelombang 2D *Shallow Water Equations* (SWE). Model SWE ini diimplementasikan secara numerik dengan menggunakan metode *Finite Volume* pada grid model *staggered*. Skema numerik yang diimplementasikan diuji untuk menyimulasikan fenomena gelombang *runup*. Dari hasil pengujian, CPU *time* dari OpenMP lebih baik dibandingkan dengan pemrograman serial ketika jumlah grid nya lebih dari 51×51. Untuk jumlah grid yang lebih kecil, arsitektur pemrograman serial memberikan waktu yang lebih efisien dibandingkan OpenMP. Selain itu, *speedup* dari OpenMP dapat mencapai 3.95 kali untuk 4 *thread* dan mencapai 6.61 kali untuk 8 *thread* lebih cepat dari pemrograman serial untuk jumlah grid 1101×1101. Efisiensi maksimum dalam simulasi ini dengan menggunakan 4 thread mencapai sekitar 91.4%, sedangkan untuk 8 thread mencapai sekitar 82.7% pada jumlah grid 1101×1101. Dapat disimpulkan bahwa skema numerik untuk model SWE 2D dengan *staggered grid* pada arsitektur OpenMP, efisiensi yang tinggi didapatkan untuk jumlah grid yang besar.

Daftar Pustaka

[1] Adytia, D. (2019). Momentum Conservative Scheme for Simulating Wave Runup and Underwater Landslide. Indonesian Journal on Computing (Indo-JC), 4(1), 29-42.

[2] Alfikri, M. Z., Adytia, D., & Subasita, N. (2019, March). Shock capturing staggered grid scheme for simulating dam-break flow and runup. In Journal of Physics: Conference Series (Vol. 1192, No. 1, p. 012041). IOP Publishing.

[3] Sri Redjeki Pudjaprasetya and Ikha Magdalena. Momentum *conservative scheme for shallow water flows*. *East Asian J. Appl. Math.(EAJAM)*, 4(2):152–165, 2014.

[4] F. Bouchut, *Nonlinear stability of finite Volume Methods for hyperbolic conservation laws: And Well-Balanced schemes for sources*. Springer Science & Business Media, 2004.

[5] E. Audusse, F. Bouchut, M. Bristeau, R. Klein, and B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. SIAM Journal on Scientic Computing, 25(6):20502065, 2004.

[6] Patrick J. Lynett, Tso-Ren Wu, Philip L. –F. Liu. Modelling Wave Runup with Depth-Integrated Equations. Coastal Engineering 46 : 89-107, 2002.

[7] G. Stelling and S. P. A. Duinmeijer. *A staggered conservative scheme for every froude number in rapidly varied shallow water flows. International Journal for Numerical Methods in Fluids*, 43(12):1329–1354, 2003.

[8] David Doyen and Putu Harry Gunawan. An explicit staggered finite volume scheme for the shallow water equations. In *Finite Volumes for Complex Applications VII-Methods and Theoretical Aspects*, pages 227–235. Springer, 2014.

[9] W. C. Thacker, Some exact solutions to the nonlinear shallow-water wave equations in Journal of Fluid Mechanics, vol. 107, pp. 499–508, 1981.

[10] O. Delestre, S. Cordier, F. James, and F. Darboux, "Simulation of rainwater overland-flow," in *12th International Conference on Hyperbolic Problems*, vol. 67. American Mathematical Society, 2008, pp. 537–546.

[11] Adytia, D., S. R. Pudjaprasetya, and D. Tarwidi. "Modeling of wave runup by using *staggered grid* scheme implementation in 1D Boussinesq model." Computational Geosciences (2019): 1-19.

[12] Glaister, P. (1988). Approximate Riemann solutions of the shallow water equations. Journal of Hydraulic Research, 26(3), 293–306.

[13] VILA, J. P., Simplified Godunov Schemes for 2x 2 Systems of Conservation Laws. SIAM J. Numer. Anal. 23, 1986. p. 1173.

[14] Zhang, S., Xia, Z., Yuan, R., & Jiang, X. (2014). Parallel computation of a dam-break flow model using OpenMP on a multi-core computer. Journal of Hydrology, 512, 126–133.

[15] Khairul Sabri, Hasbi Rabbani, Putu Harry Gunawan. *OpenMP Performance for Benchmark 2D Shallow Water Equations Using LBM*. Telkom University, *School of Computing*, Bandung. 2018

[16] P.H. Gunawan, S. Juliati, M. R. Pahlevi, D. Adytia. Openmp and MPI Architectures for Simulating 1D Water Oscillation on Parabolic Domain. In *International Journal of Engineering & Technology*, 8 (1.9) (2019) 230-236

[17] P. H. Gunawan, "Scientific parallel computing for 1d heat diffusion problem based on openmp," in *Information and Communication Technology(ICoICT), 2016 4th International Conference on*. IEEE, 2016, pp. 1–5.