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.