# Dokumentace PPP projektu

Lukáš Plevač < *xpleva07@vutbr.cz* >

## 1 Úvod

Tento dokument ukazuje a popisuje chování implementovaného algoritmu pro výpočet šíření tepla ve 2D prostoru. Algoritmus byl implementován v C++ s knihovnami MPI a HDF5. Algoritmus má dva režimi předávání dat mezi procesy a to pomocí přímého přístupu do paměti (RMA) a klasické pear to pear komunikace (P2P). Pro potřeby dekompozice prostoru je možné použít 1D dekompozici (podle osy X) nebo 2D dekompozici. Algoritmus je možné také použít v hybridním režimu kdy část zdrojů spravuje MPI a část OMP. Následující kapetoly ukáží grafy efektivity, silného škálování a zrychlení.

## 2 Grafy

Grafy jsou ve formátu SVG a aby bylo možné je dobře číst je nutné si je přiblížit. Malá velikost grafů byla zvolena aby se vešly do dvoustránkového dokumentu.

### 2.1 P2P, 2D decomp, No hybrid

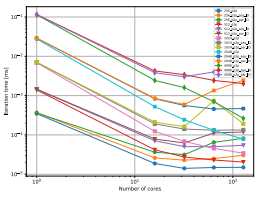

Figure 1: Silné škálování

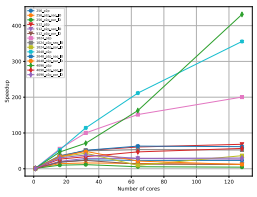

Figure 2: Zrychlení

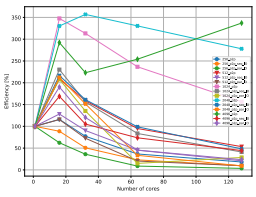

Figure 3: Efektivita

## 2.2 P2P, 1D decomp, hybrid

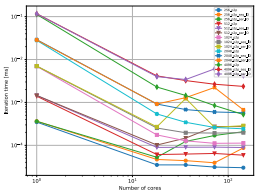

Figure 4: Silné škálování

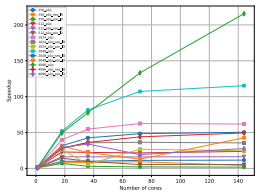

Figure 5: Zrychlení

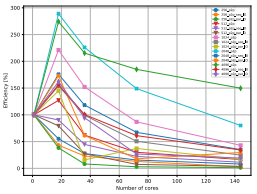

Figure 6: Efektivita

## 2.3 RMA, 2D decomp, No hybrid

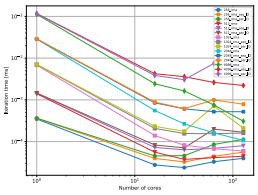

Figure 7: Silné škálování

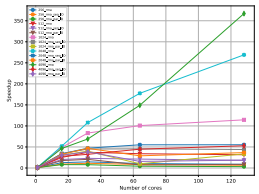

Figure 8: Zrychlení

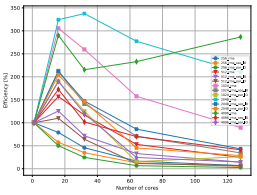

Figure 9: Efektivita

## 2.4 RMA, 1D decomp, hybrid

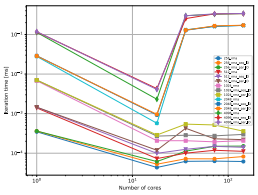

Figure 10: Silné škálování

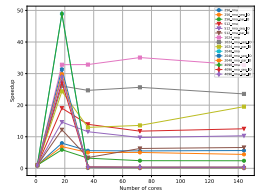

Figure 11: Zrychlení

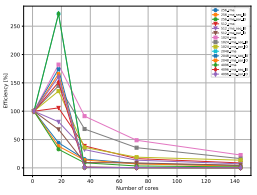

Figure 12: Efektivita

## 2.5 RMA, 2D decomp, hybrid

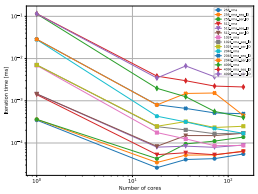

Figure 13: Silné škálování

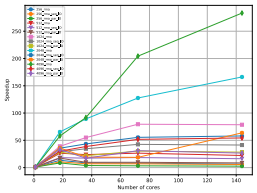

Figure 14: Zrychlení

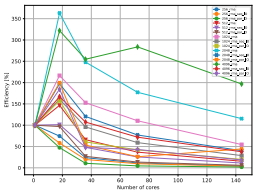

Figure 15: Efektivita

## 2.6 P2P, 2D decomp, hybrid

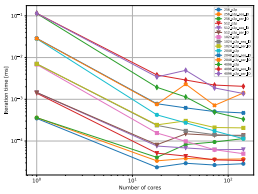

Figure 16: Silné škálování

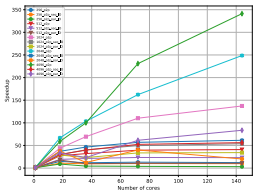

Figure 17: Zrychlení

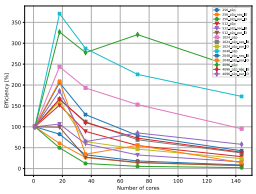

Figure 18: Efektivita

## 2.7 Vysoká efektivita

Efektivita pro parallení kód je větší než 100% což by nemělo nastat, ale v tomto případě je to vpořádku. Tento jev byl způsoben tím že paralení verze pouzívá při výpočtu SIMD jednotky, narozdíl od sekvenční verze která je nepoužívá, toto vede k tomu že paralení kód má větší efektivity než původní sekveční kód a proto efektivita roste nad 100%.

## 2.8 Rozdíl mezi škálováním 1D a 2D dekompozice

2D dekompozice se podle grafů škáluje lépe. Tento jev je způsoben že u 2D dekompozice je větší vnitřní oblast kterou můžeme počítat a překrýt tím odesílání HALO oblastí. Dále při 1D dekompozici se dříve dostaneme na malé bloky které mají vetší HALO oblasti než vnitřní výpočetní část což vede na obrovské nároky na komunikaci a dlouhé čekání na výsledky komunikace. Toto se dá (a bylo) snadno ověřit přidáním časovače který bude počítat dobu paraleního nezávislého výpočtu a dobu po kterou se čekalo na HALO zóny.

## 2.9 Paralelní IO

Ikdyž bylo předpokládáno že paralení IO bude efektivnější než sekvenční bohužel tomu tak nebylo. V některých případech je pralelní IO dokonce horší než sekvenční IO. Tento jev je způsobem nutnosti připaravit data pro zápis do souboru a častým bojem o zápis do souboru mezi prcesy. Toto by šlo dobudoucna vyřešit vetším laděním souběžného zápisu do souboru.

## 2.10 Přínos překrytí komunikace a výpočtu

Smyslem je aby se čas kdy se čeká na data od sousedů smysluplně využil. Pokud bychm nejdříve vše vypočítaly a pak hned poslaly trvalo by to déle než když výpočet s komunikací překryjeme tak že nejdříve vypočítáme okrajové oblasti ty necháme odeslat a pak začneme počítat střed. tím doje překrytí počítaání středu a přeposlání okrajových oblastí mezi sousedy a díky tomu procesor nebude zbytečne stát a čekat. Toto jde ověři tím že přidáme timer před wait all a za wait all a uvidíme že pokud překryjeme komunikaci výpočtem budeme čekat méně času. Samozřemě záleží jak je naše středová oblast velká pokud je moc malá nebudeme mít dostatek výpočtu na překrytí komunikace, jako když máme mnohem větší středovou část to je také důvod proč program s vetším vstupem lépe škáluje.

## 2.11 Slabé škálování

Grafy slabého škálování jsou obsaženy v grafech silného škálování stačí pouze vzdy zvětšit úlohu se zvetšujícím se počtem procesorů, tedy stačí vždy se posunout o jeden bod v grafu a změnit velikost vstupu 2x.