Concurrent

# Gemaakt door:

Joeri de Wagt

Koen elderhorst

[Gemaakt door: 1](#_Toc31403476)

[Inleiding 3](#_Toc31403477)

[Probleemstelling 3](#_Toc31403478)

[Resultaten 3](#_Toc31403479)

[Conclusie 5](#_Toc31403480)

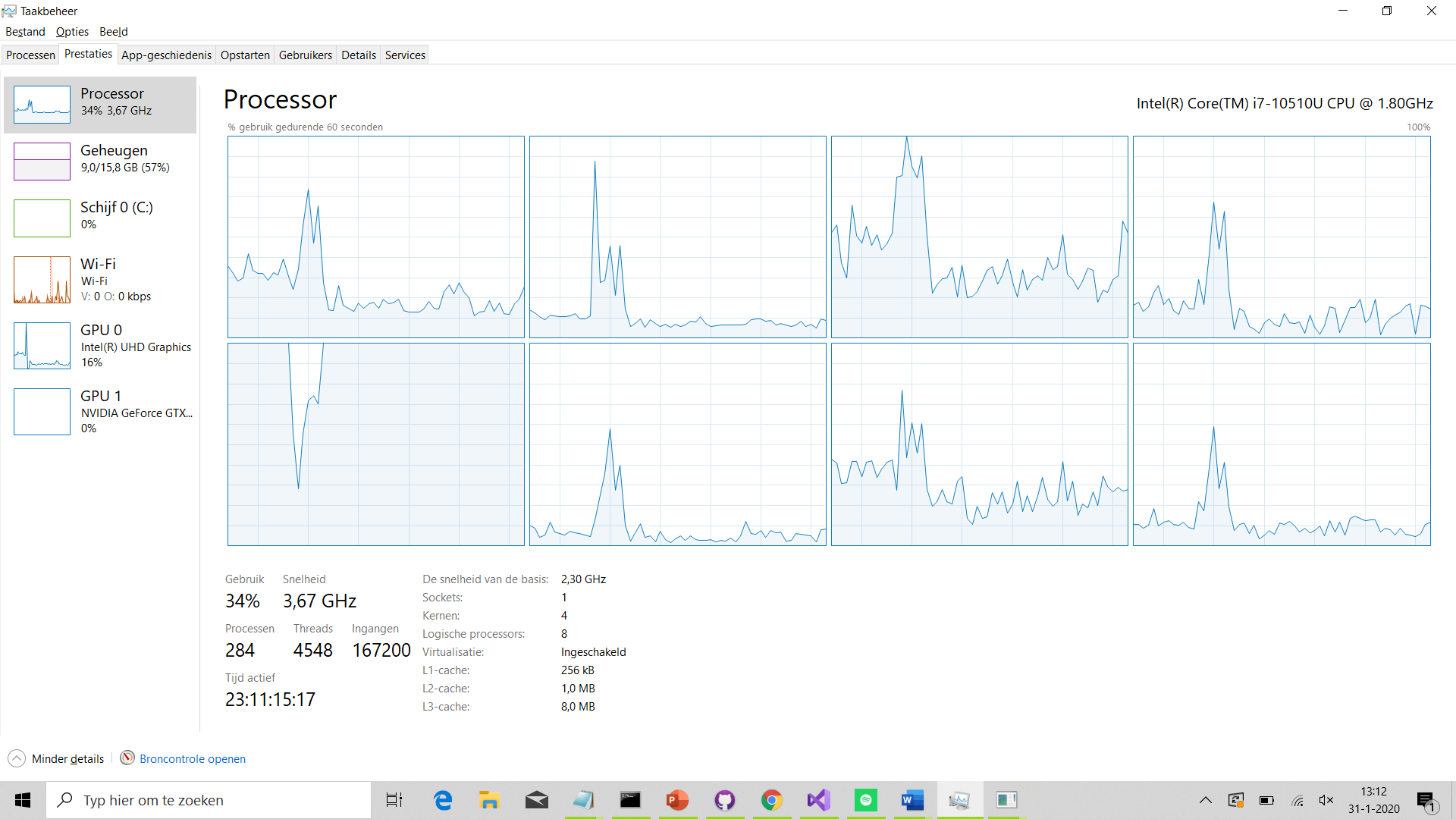
# 

# 

# Inleiding

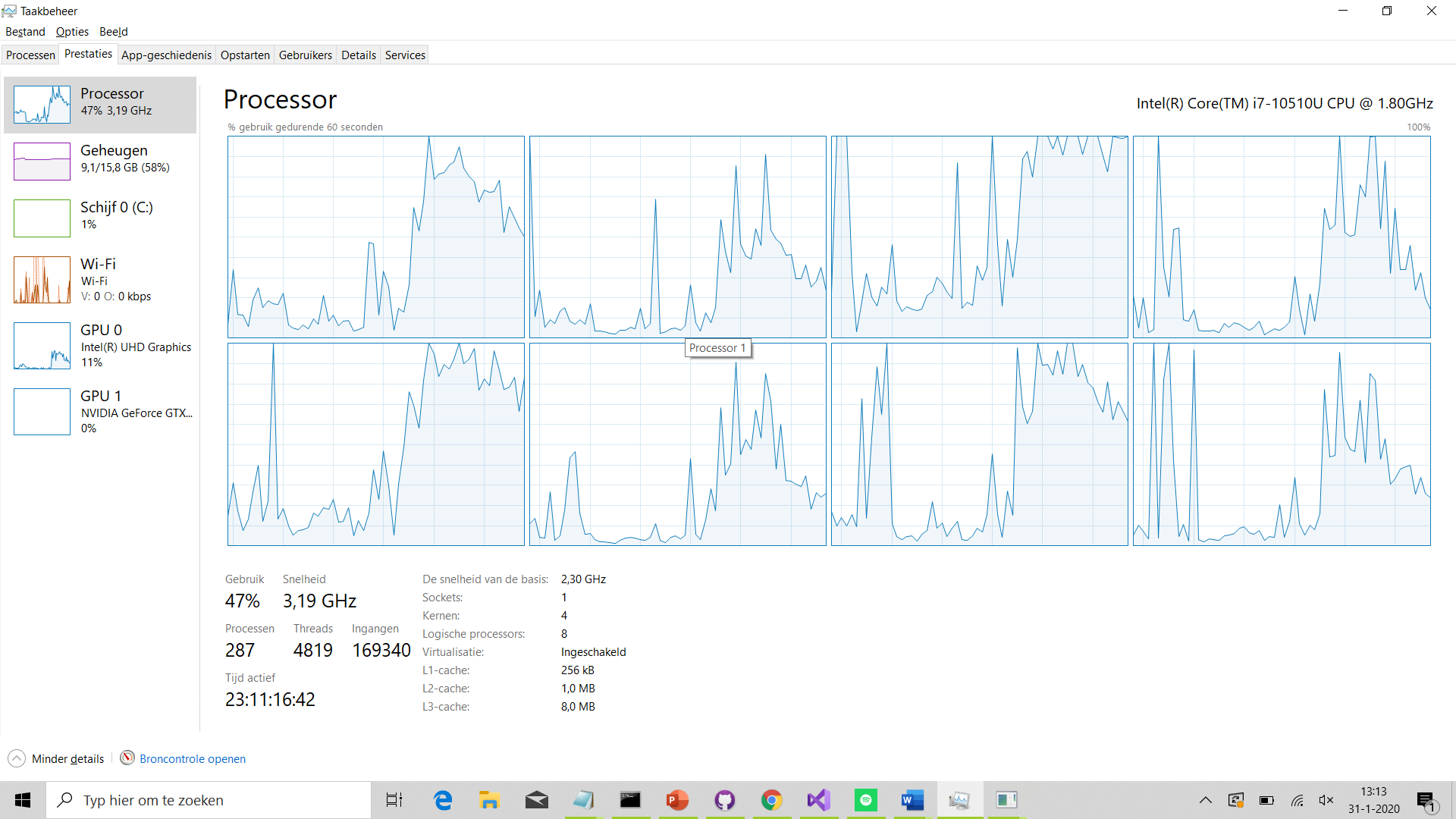
Voor het project deze periode hebben wij de opdracht gekregen om een oorlogssimulatie te optimaliseren, zodat hij zo snel mogelijk is. Dit hebben we gedaan door algoritmes te vervangen of aan te passen, en door gebruik te maken van *threads*. In dit verslag staan de veranderingen door middel van *Threads* gedocumenteerd. Dit is gedaan door middel van *screenshots* en uitleg van de veranderingen.

# Probleemstelling

Om te beginnen hebben wij eerst het probleem vastgesteld. Het probleem van de applicatie is dat de simulatie op één core draait, wat niet optimaal is. Door eerst dit met zekerheid vast te stellen konden wij beginnen met een doel te stellen.

Nu we weten dat de applicate bijna volledig op één core draait, is voor ons duidelijk wat wij aan moeten passen. Namelijk het invoegen van *Threads*, dit wordt gedaan door middel van een *threadpool (een design pattern waarin meerdere threads* zijn opgeslagen).

# Resultaten

Zoals in de afbeelding hieronder te zien is, wordt de workload beter verdeeld over de cores en worden calls naar de functies nu parallel uitgevoerd. Door de veel aangestuurde functies zoals Update en ParticleBeams te te laten verwerken door threadpool. Door niet alles maar te laten queuen verdelen we de work naar alleen noodzakelijke functies. De span van de thread stopt zodra de functie out of scope raakt, aangezien wij functies aanroepen die elke frame een taak queuen is de span van de treads kort maar heel frequent. 

Een voorbeeld van de versnelling die geleverd kan worden wordt hieronder getoond aan de hand van afbeeldingen. In dit geval zijn de theads toegevoegd aan het stuk code dat checkt of een raket een target raakt.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Zoals te zien is in de afbeeldingen hierboven heeft de thread voor een speedup van één seconde gezorgd. Dit lijkt niet veel op het eerste gezicht, maar bedenk wel dat er 2558 tanks zijn die elk zo vaak mogelijk schiet.



Figuur eindproduct

# Conclusie

Als eindconclusie kunnen wij zeggen dat na het bepalen waar de zwakte lag in rekenkracht. Hebben dit kunnen verhelpen door taken sequentieel te laten verwerken door de cores met behulp van een threadpool. wat de optimalisatie zeer heeft verhoogd.