\chapter[Run time Comparison]{Run-Time Comparison of fully Parallel Algorithm versus only Basilar Membrane Velocity Parallelized and Full Serial Algorithm}

\label{sec:results}

\section{Run Times for Different Configurations}

\label{sec:run-time-configurations}

Table \ref{tab:time-analysis} compares several modes of operations and run time for three system configurations.

\begin{enumerate}

\item \citealt{Barzelay2011} simulated the cochlear model on a CPU without the using a GPU. The model required $1.36\times10^{12}$ operations per second of input signal \cite{Saboddd2013682}, making this system unfeasible for real-time diagnostics for audiologists. Some of the time measurements extrapolated from smaller runs because of the impossibility of waiting years for completion. However, this system is useful for verifying our program for certain inputs.

\item \citealt{Saboddoron20131215} improved significantly the calculation speed of BMV, which is the most resource-intense part of the algorithm. By copying the results to CPU memory, saving to a HD, and loading it with Matlab, we can calculate the \ac{anr} response. Although this approach is superior to the first, it has weaknesses: copying data from the GPU to CPU and then to the HD takes time, as does the necessity of reloading data from the HD. Computing \ac{anr} on a CPU is also much slower than doing so on a GPU. Our work in \cref{sec:cochlear\_model\_optimization}shows that, because of an error in calculating the signal input time with the desired precision, the output is not useful for calculating \ac{anr} and, concomitantly, the \ac{jnd}. Thus, after the first 80~ms, the parallelism significantly degrades.

\item The present work improves the speed of generating databases for diagnostics by observing that the input for each damage profile is small (a few kB at most) for OHC and IHC damage across the cochlea, and signal power and frequencies for testing and output also require only a few kB for an audiogram at various levels of input. Thus, by implementing the calculation of the BMV + \ac{anr} + \ac{jnd} on a GPU, we avoid the expensive operations of copying large amounts of input BMV data to disk and parallelizing \ac{anr} and \ac{jnd}. This is allowed by setting the amount of allocated memory on the GPU at run time and calculating the time so that precision remains constant.

\end{enumerate}

\phantomsection

\subsection{Comparing run-time for different tasks}

\label{sec:tasks-comparing-run-time}

To examine the run time of our program, we compare it to two similar programs for several typical tasks. The entire CPU algorithm was developed by \cite{Barzelay2011} and, for the CPU-GPU combination, we followed the work of \citealt{Saboddd2013682} and added ANR and JND calculations to the results in Matlab. The timing for each task is shown in \Cref{tab:time-analysis}. The acceleration factors differ for several reasons, with the most significant improvement being the generation of hearing-impairment databases by the program \cite{Saboddoron20131215}. Several run times for the pure CPU were approximated because of the very long time required to run them (days for $\cM$, years for $\cC$). We also verified that the run time was linear in the number of profiles results, as expected from the serial nature of the \ac{cpu}.

\begin{enumerate}

\item $\cA$: Calculating an audiogram for a healthy patient by testing single-input power (adjusted from 7.5 to 50 dB due to lower \ac{splref}) \cite{Barzelay2011} for four noise levels.

\item $\cB$: Similar to $\cA$, but with 10 levels of signal power to implement \cite{odedst2017} the minimum-required-power method. Note that are our program execution time multiplied by less than five because the GPU was under-utilized in $\cA$ since the profile signals were too short to fully utilize the GPU.

\item $\cC$: To create the database we need profiles of damage at small intervals, 1\% for \ac{ohc} and \ac{ihc} gives 10\,000 profiles to test.

\item $\cD$: Examine the effect of single prescription for the Hebrew word ``bor,’’ of length 0.8 s. For the pure-CPU method, the processing time is similar to that for a full damage profile. The pure-CPU approach preserves the relation of ~1000 s for every input signal because the processing time dominates every task. For other approaches and short tasks copying data to \ac{cpu} for graph presentation on Matlab requires more time.

\item $\cE$: Simulate \ac{anr} for multiple words spoken by both male and female voices for single-prescription method. Total signal length is 10.3 s. Note that our method loses lots of its relative improvement because of Matlab rendering time.

\item $\cL$: \ac{anr} and \ac{jnd} calculation time for $\cB$, since both are done on a CPU, a pure-CPU method can work with 40~ms of input, but a reliance on an erroneous GPU calculation demands four times more data. The input length for tasks $\cB$ and $\cL$ is 45.44~s for combined method and 11.4~s for the other methods.

\item $\cM$: Test the improvement in JND as a function of signal time, as detailed in \ref{sec:additional-experiment}. Note that we approximate the run time on a pure CPU by multiplying $\frac{\#profiles}{4}$ by the amount of run-time required to test four profiles at $\cA$.

\end{enumerate}

\begin{table}%

%\tbl{Processor configuration for sequential, single-core computation.\label{tab:CPU-preferences}}{%

\begin{tabular}{|p{15mm}|p{22mm}|p{45mm}|p{45mm}|}

\hline

\diaghead(-2,1){taskMethod}%

{task}{Method}& Pure CPU & BM Velocity on GPU, \ac{anr} and \ac{jnd} on CPU & Full GPU process \\\hline

$\cA$ & 1150 & 93 & 0.31 \\\hline

$\cB$ & 11500 & 928 & 1.35 \\\hline

$\cC$ & $1.15\times 10^8$ & $9.28\times 10^6$ & $1.35\times 10^4$ \\\hline

$\cD$ & 811 & 13 & 0.66 \\\hline

$\cE$ & 10180 & 123 & 20.9 \\\hline

$\cL$ & 35 & 135 & 0.11 \\\hline

$\cM$ & $6\times10^6$ & 60000 & 892 \\\hline

\end{tabular}

\caption{Comparing run time for different tasks: with CPU only, GPU + CPU. All heavy calculation tasks on GPU are measured in seconds. }

\label{tab:time-analysis}

\end{table}%

We tested the pure-CPU version of the algorithm on the single core described in \cref{tab:CPU-preferences}

For the GPU implementation, we ran the simulations on an NVidia GTX1080Ti.

\Cref{tab:GPU-preferences} shows the configuration of the cards.

\begin{enumerate}

\item By fixing the linear interpolation between two input samples to ensure time invariance, as described in \cref{sec:single-precision-computations-updates}, the previous program could process 0.3~s in each execution but only the first 80~ms could be used by \cite{odedst2017}, because each block interval handles 20~ms of inputs. Thus, the card could only compute four block intervals in a single kernel run. We show that, with the optimal occupancy division of 112 intervals (28 SMs, each executing four blocks), the use of the GPU improved performance by a factor of 28 for all tasks.

\item To compensate for previous program calculation error, the JND calculations required averaging 160~ms of input. We show, however, that using less than 40~ms and the multiplication factor of Section~\ref{sec:nsight-for-congestion} improves performance by a factor of four, but only for database generation.

\item Both previous works used to calculate \ac{anr} on the CPU required copying the BMV to memory and then to the HD, plus a slow calculation on the CPU because they uploaded the frequencies, powers, noise power, and damage profile (which is few kB) to the GPU and, at the end of JND calculation, downloaded from the GPU memory.

\end{enumerate}

\section{Acceleration due to Work-Flow Modifications}

\label{sec:work-flow-mods-accelarations}

We examined in \cref{sec:execution-flow-optimization} changes to the work flow that allow us to avoid both memory allocations and releases (except for a limited number of times). We can see the run time saved for each run in \cref{fig:memory-size-handling-times}; larger buffers make this difference more significant.

We also replaced reading and writing to the HD by RAM by using the Matlab Application Programming Interface, which save time as shown in \cref{fig:memory-size-allocation} and is also relative to buffer size.

\begin{figure}

\pgfplotstableread[col sep=comma,header=true,columns={Array Size,Output Time HD write,Output Time Array Copy,Accelaration}]{allocatingTimes.csv}\datatable

\pgfplotstableread[col sep=comma,header=true,columns={Array Size}]{allocatingTimes.csv}\datarow

\pgfplotsset{every x tick label/.append style={font=\small}}

\pgfplotsset{every y tick label/.append style={font=\small}}

\setlength{\subfigcapmargin}{.1in}

\centering

\begin{tikzpicture}

\begin{axis}[

xlabel={Buffer Size in MB}

,ylabel={Output Copy Time, Seconds}

,ylabel style={align=center,text width=6cm,color=blue}

,xticklabels from table={\datarow}{Array Size}

,width = 0.85\*\textwidth

,xminorticks=false

,xtick distance=1

,xtick=data

]% table

\addplot[blue] table[x expr=\coordindex,col sep=comma,y=Output Time HD write] {allocatingTimes.csv};

\addplot[red] table[x expr=\coordindex,col sep=comma,y=Output Time Array Copy] {allocatingTimes.csv};

\legend{HD Read and Write time,RAM Read and Write time}

\end{axis};

\begin{axis}[

,ylabel style={align=center,text width=6cm,color=purple}

,ylabel={Relation between copy and read time from HD and RAM}

,hide x axis

,nodes near coords

,width = 0.85\*\textwidth

,axis y line\*=right

,point meta=y

]% table

% \pgfplotstablegetcolsof{\datatable};

% \pgfmathtruncatemacro\numberofcols{\pgfplotsretval-1}

%\pgfplotsforeachungrouped \i in {Accelaration}{

\addplot[purple,mark=\*] table[x expr=\coordindex,col sep=comma,y=Accelaration] {allocatingTimes.csv};

%};

\end{axis};

\end{tikzpicture}

\caption{Comparison of output copy time, HD, and RAM for different buffer sizes. The left axis shows the combined time of the write for the program and the read of the results by Matlab. The right axis shows the relation between the times for HD and RAM access.}

\label{fig:memory-size-handling-times}

\end{figure}

\begin{figure}

\pgfplotstableread[col sep=comma,header=true,columns={Array Size,Time for memory release,Time To allocate memory}]{allocatingTimes.csv}\datatable

\pgfplotstableread[col sep=comma,header=true,columns={Array Size}]{allocatingTimes.csv}\datarow

\pgfplotsset{every x tick label/.append style={font=\small}}

\pgfplotsset{every y tick label/.append style={font=\small}}

\setlength{\subfigcapmargin}{.1in}

\centering

\begin{tikzpicture}

\begin{axis}[

xlabel={Buffer Size in MB}

,ylabel={Allocating and eleasing Buffers, Seconds}

,ylabel style={align=center,text width=6cm,color=blue}

,xticklabels from table={\datarow}{Array Size}

,width = 0.85\*\textwidth

,xminorticks=false

,xtick distance=1

,xtick=data

]% table

\addplot[blue] table[x expr=\coordindex,col sep=comma,y=Time To allocate memory] {allocatingTimes.csv};

\addplot[red] table[x expr=\coordindex,col sep=comma,y=Time for memory release] {allocatingTimes.csv};

\addplot[purple] table[x expr=\coordindex,col sep=comma,y expr={\thisrow{Time To allocate memory}+\thisrow{Time for memory release}}] {allocatingTimes.csv};

\legend{Allocating Buffers Time, Releaseing Buffers Time, Summary}

\end{axis};

\end{tikzpicture}

\caption{Comparison of output copy time, HD, and RAM for different buffer sizes. The left axis shows the combined write time for the program and read of the results by Matlab. The right axis shows the relation between the times for HD and RAM access. }

\label{fig:memory-size-allocation}

\end{figure}

\section{Additional Experiment}

\label{sec:additional-experiment}

This improvement in speed allowed us to conduct many more experiments in a shorter time than with the old programs (see examples in \cref{fig:jnd\_by\_time}).

We examined the model assumption that the calculation of the JND by using the Cram\’er-Rao lower bound with the \ac{rms} method \cite{furst2015} approximates the JND for every signal length.

However, the JND improves as a function of time far beyond that of a \ac{nh} person, so we added the factor $k\_{JND}$ described in \cref{sec:jnd-single-tone-optimization} to compensate.

\begin{figure}

\centering

\includegraphics[width=0.95\textwidth,keepaspectratio=true]{figs/jndByTime}

\vspace{.2in}

\caption{Test of JND for healthy patient as a function of signal time. The JND for each tested tone is measured as function of time interval. Twenty-two intervals of 100 to 500~ms were used, with steps of 100~ms, and 1 to 9~s with steps of 0.5~s. Ten power levels (from 10 to 100~dB) were used for seven frequencies. The total was 6055~s of audio. The times are shown in \ref{tab:time-analysis} for $\cM$. }

\label{fig:jnd\_by\_time}

\end{figure}